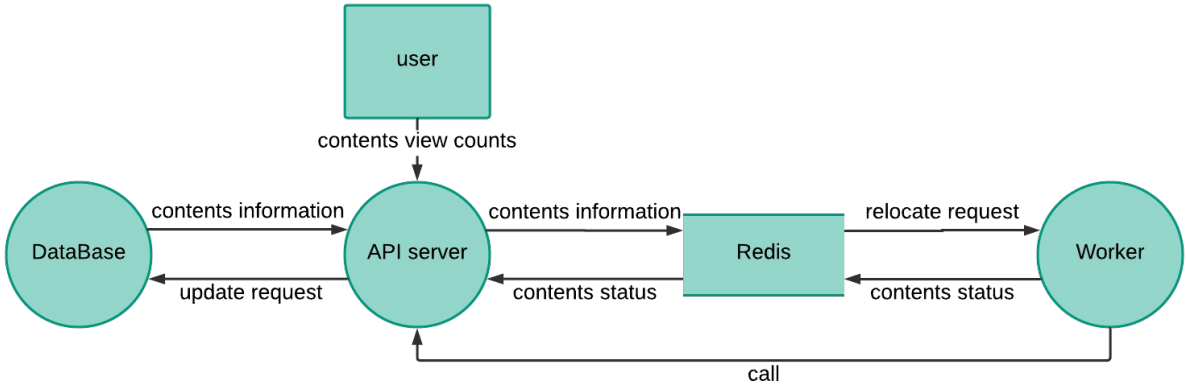
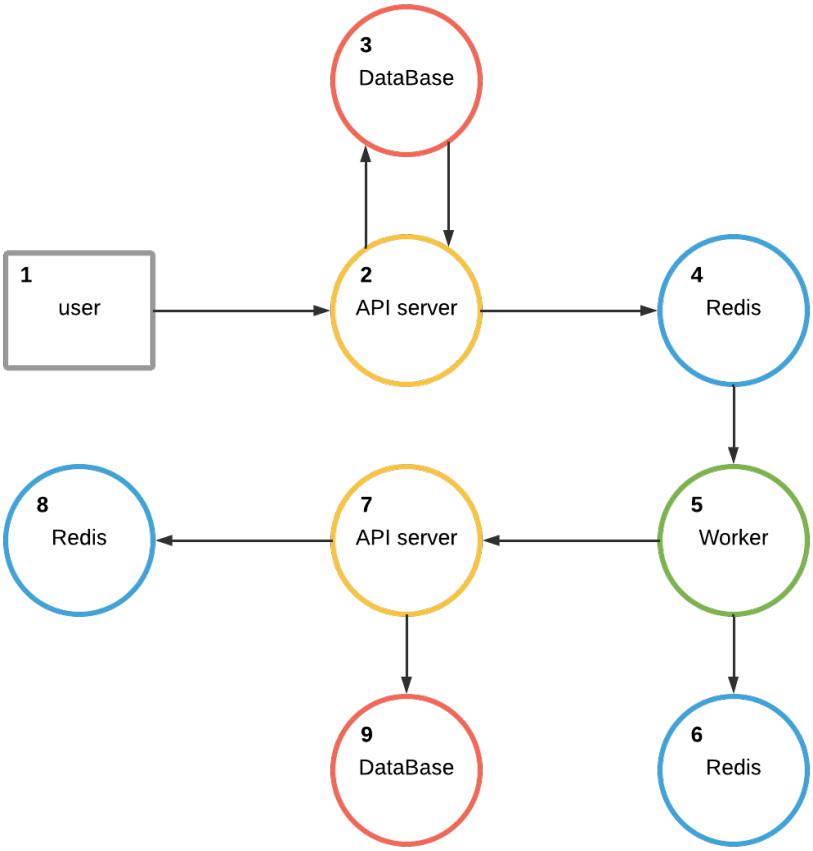
INI Intern Final Project

장예훈 (DB 구축 및 쿼리 모듈화) 18.07.27 Tue - 18.08.02 Thu

DFD



Work Flow



User (1)

- api 서버에 cid와 count 정보를 post method로 전달

API 서버 (2, 7)

- user로부터 cid와 count 정보를 받아 쿼리 모듈을 이용하여 db에서 필요한 정보를 쿼리 한 후 content의 level relocate 정보에 따라 redis를 업데이트
- worker의 호출을 받으면 redis에서 content status가 'done' 인지 확인 후 content의 relocate 된 level에 맞게 쿼리 모듈을 이용해 db 업데이트

DataBase (3, 9)

- api 서버에서 인자를 db 쿼리 함수에 넘기면 해당 프로젝트와 연결된 db 쿼리를 실행

Worker (5)

- redis의 content status를 확인하여 status가 'update' 인 content의 relocation을 진행
- relocate가 정상적으로 완료된 content에 대하여 redis의 content status를 'done' 으로 바꾼 후 api 서버를 호출

Redis (4, 6, 8)

- content count가 업데이트 될 때 마다 올바른 level에 relocate 되도록 content에 대한 정보를 실시간으로 api 서버와 worker에 제공 (사실상, api 서버와 worker가 실시간으로 redis를 확인)

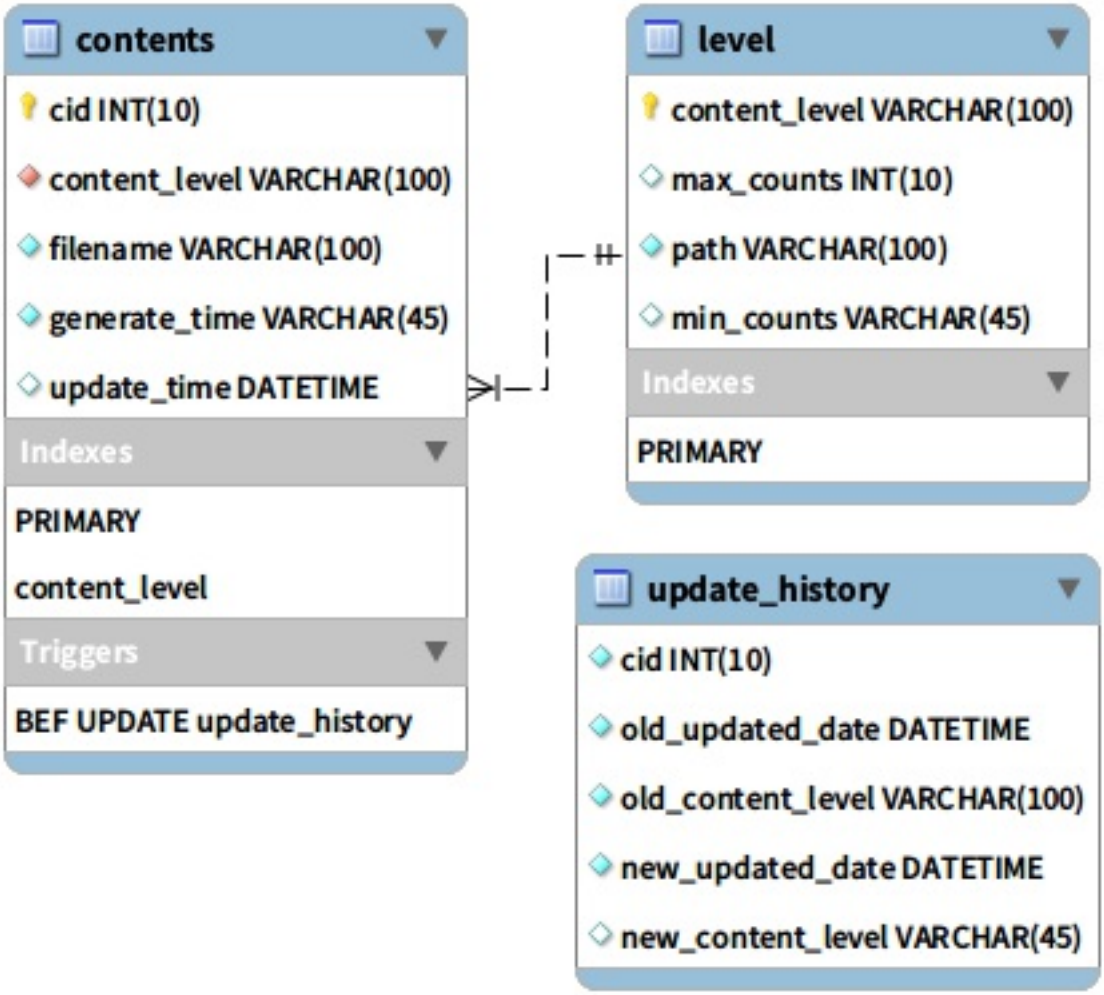
개발 환경

- Ubuntu 16.04.5 LTS
- Python 3.5
- MySQL 5.7.23
- PyMySQL 0.9.2
- Anaconda 4.3.11

DB Process

DB 구축

1. ERD



2. DB 테이블

level

content의 view count 수에 따른 level 정보를 저장하는 테이블

- `content_level` : level의 이름
- `max_counts` : 해당 level의 maximum count
- `min_counts` : 해당 level의 minimum count
- `path` : 해당 level의 content가 저장 될 절대 경로

contents

content의 정보를 저장하는 테이블

- `cid` : content를 구분하는 id
- `content_level` : content의 현재 level, level 테이블의 `content_level` 을 참조
- `filename` : content의 filename, worker가 relocate 할 때 경로 설정을 위해 필요
- `generate_time` : content 최초 생성 시간

- `update_time` : content의 level relocate 시간

update_history

contents level 정보가 업데이트 될 때마다 update history를 저장하는 테이블로 contents 테이블의 `content_level`에 업데이트 이벤트가 발생하면 데이터를 insert 하는 트리거와 연결

- `cid` : 업데이트 된 content의 id
- `old_updated_date` : content level의 현재 업데이트 이전 업데이트 시점
- `old_content_level` : content level의 현재 업데이트 이전 content level
- `new_updated_date` : content level의 현재 업데이트 시점
- `new_content_level` : content level의 현재 업데이트 content level

3. DB 트리거

- contents 테이블의 업데이트 이벤트가 발생하면 트리거가 작동되고 update_history에 업데이트 시점과 업데이트 된 content level이 기록됨
- 트리거
  `sql CREATE DEFINER = `root`@`localhost` trigger update_history before update on contents for each row begin insert into update_history values(old.cid, old.update_time, old.content_level, now(), new.content_level); end`

DB 쿼리 클래스 생성

프로세스 내에서 필요한 db 쿼리 클래스 생성

1. 클래스 구조

```
class db:
  def __init__(self):
    ...

  def select(self, table, column, where_clause=None, order_by=None):
    ...

  def insert_contents(self, table, cid, file_name):
    ...

  def update_level(self, cid, content_level):
    ...
```

2. 함수 설명

1. __init__

- db 클래스 생성 시 `PyMySQL`로 MySQL 연결할 때 필요한 파라미터 설정
- `pymysql.connect()`를 이용하여 MySQL 연결

2. select

- api 서버와 db 간 데이터 상호 교환 시 반복 사용되는 MySQL select 쿼리문을 모듈화 한 것
- `table`, `column`, `where_clause`, `order_by`가 함수 인자이며 이 중 필수 인자는 `table`과 `column`

3. insert_contents

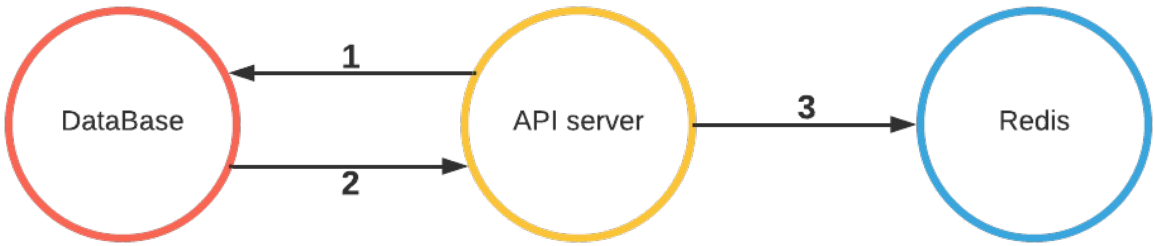
- content가 새로 업로드 될 때 콘텐츠에 대한 정보를 MySQL contents 테이블에 insert 해주는 기능을 모듈화 한 것
- min_count가 0인 level이 default `level`
- `generate time`은 insert 쿼리문 실행 시점이 적용되므로 본 함수는 콘텐츠가 업로드 됨과 동시에 실행되어야 함

4. update_level

- api 서버가 본 함수를 호출하면 MySQL contents 테이블에서 content_level을 update 하는 쿼리문을 모듈화 한 것
- update 하고자 하는 콘텐츠의 `cid`와 relocate 된 `contet_level`이 필수 인자
- 본 함수가 실행되면 contents table에서 content_level과 update_time column이 업데이트 이전에 update_history 테이블과 연결된 트리거가 먼저 작동

프로세스 상세 및 실행 결과

redis에 content 정보 업데이트



1. api 서버 - db 쿼리 함수 호출

- api 서버는 content의 현재 위치 level과 count가 해당되는 범위의 위치 level 비교를 위해 쿼리 모듈로 db에서 데이터를 쿼리 함

- content 테이블에서 user로부터 받은 cid로 해당 content의 현재 위치 level을 쿼리 하는 `select 함수` 호출
- `select 함수`로 level 테이블의 모든 coulmn을 쿼리 한 후 user로부터 받은 count와 비교 연산하여 target level을 반환함

2. db - 쿼리 결과 api 서버에 반환

- db는 `select 함수`로 쿼리 된 결과를 api 서버에 반환함
  - MySQL content 테이블 내의 cid 1 정보

#	cid	content_level	filename	generate_time	update_time
1	1	gold	a.mp4	2018-07-29 15:31:24	2018-08-02 18:30:17

  - api 서버에서 `select 함수`를 호출하여 cid가 1인 content의 level 추출하는 함수 결과

```
#python shell
\>>> import redis_encode as re
\>>> re.get_level_from_db(1)
{'content_level': 'gold', 'file_name': 'a.mp4'}
```

  - MySQL level 테이블

#	content_level	max_counts	path	min_counts
1	bronze	999	/etc/inisoft/redis_project/bronze/	0
2	gold	10000000	/home/inisoft/workspace/inisoft/redis_project/gold/	2000
3	silver	1999	/var/www/html/redis_project/silver/	1000

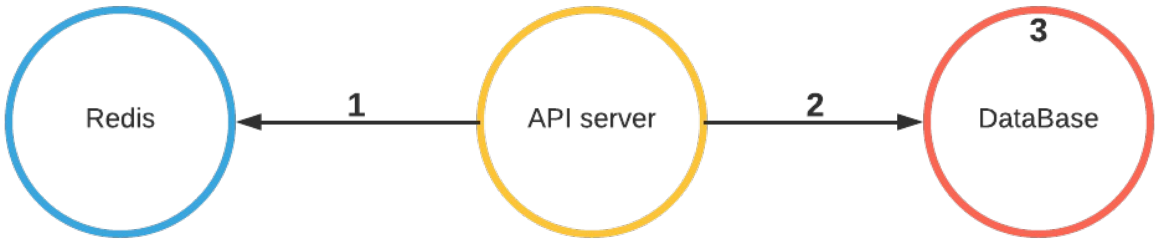
  - api 서버에서 `select 함수`로 level 테이블을 쿼리 한 후 각 level의 count와 countent의 count를 비교 연산하여 target level을 반환하는 함수 결과

```
#python shell
\>>> import redis_encode as re
\>>> re.get_target(2342)
'gold'
\>>> re.get_target(1548)
'silver'
\>>> re.get_target(356)
'bronze'
```

3. api 서버 - redis에 content에 대한 정보 업데이트

- db에서 쿼리 한 데이터를 알맞게 처리한 후 redis에 content 정보를 업데이트

relocate가 완료된 content 정보 업데이트



1. api 서버 - redis 업데이트

- content의 relocate를 마친 worker의 호출을 받은 api 서버는 해당 cid에 대해 redis의 content status가 'done' 인지 확인

2. api 서버 - db 업데이트

- redis의 content status가 'done' 인 것을 확인 한 api 서버가 쿼리 모듈로 db에서 content status를 업데이트 하는 `update_level 함수`를 호출
  - `update_level 함수` 호출 전 MySQL content 테이블 내의 cid 12 정보

#	cid	content_level	filename	generate_time	update_time
12	12	silver	l.mp4	2018-07-29 15:31:27	2018-08-02 18:30:19

  - api 서버에서 `update_level 함수` 호출 결과

```
\>>> import redis_encode as re
\>>> re.update_db_level(12, 'gold')
1
```

  - `update_level 함수` 호출 후 MySQL content 테이블 내의 cid 12 정보

#	cid	content_level	filename	generate_time	update_time
12	12	gold	l.mp4	2018-07-29 15:31:27	2018-08-07 14:27:44

3. db 업데이트

- api 서버의 쿼리 모듈 호출로 content 테이블이 업데이트 되면 db에 `insert 트리거`가 작동하여 content 테이블 업데이트 시점에 update_history 테이블에 row가 추가됨
  - MySQL content 테이블의 `insert 트리거` 작동 전 cid 12 의 update_history 테이블

#	cid	old_updated_date	old_content_level	new_updated_date	new_content_level
1	12	2018-05-01 12:32:45	gold	2018-07-26 11:26:18	silver
2	12	2018-07-26 11:26:18	silver	2018-08-01 17:55:23	gold
3	12	2018-08-01 17:55:23	gold	2018-08-02 18:30:19	silver
4	12	2018-08-02 18:30:19	silver	2018-08-06 12:34:08	silver

- MySQL content 테이블의 `insert 트리거` 작동 후 cid 12 의 update_history 테이블

#	cid	old_updated_date	old_content_level	new_updated_date	new_content_level
1	12	2018-05-01 12:32:45	gold	2018-07-26 11:26:18	silver
2	12	2018-07-26 11:26:18	silver	2018-08-01 17:55:23	gold
3	12	2018-08-01 17:55:23	gold	2018-08-02 18:30:19	silver
4	12	2018-08-02 18:30:19	silver	2018-08-06 12:34:08	silver
5	12	2018-08-02 18:30:19	silver	2018-08-07 14:27:44	gold