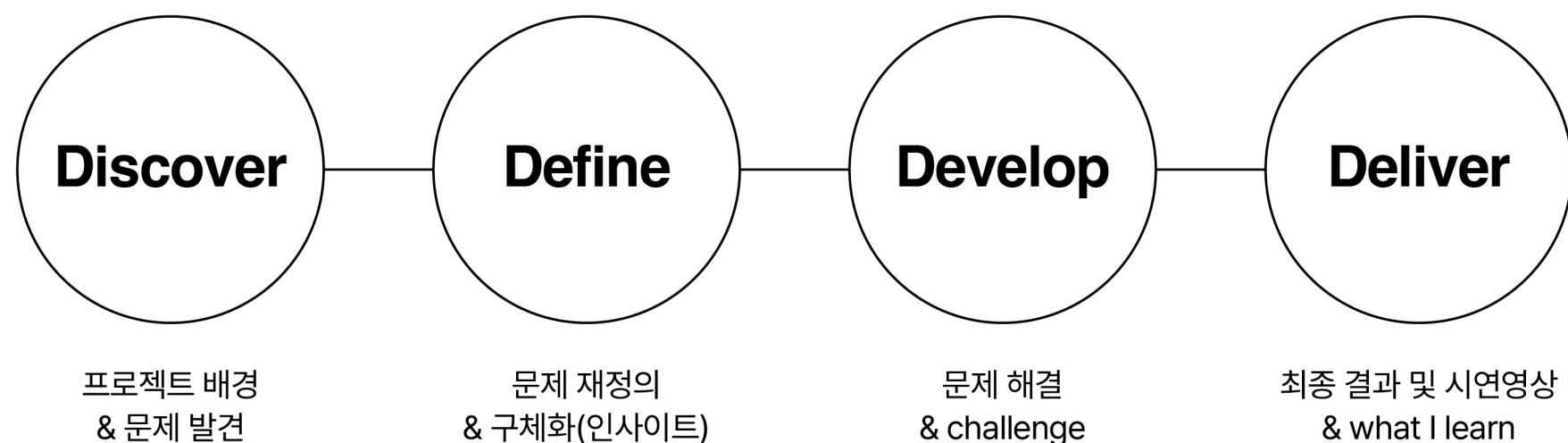


넷플릭스 VFX 아카데미 TD 개인 프로젝트 발표

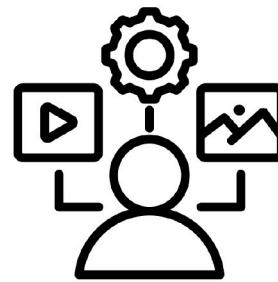
# Version Pinning

3기 교육생 고예은

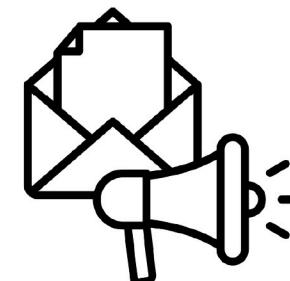
# Contents



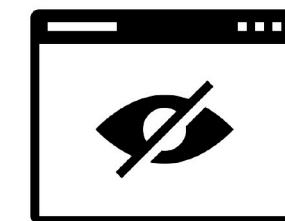
## 프로젝트 배경 | background



예시를 들어보자면,  
텍스처팀(T)이 모델링팀(M)에게 받은  
버전1 파일로 작업을 진행하고 있습니다.



이때, 모델링팀에서 버전2 파일을  
업로드하면 텍스처팀은 그 사실을  
확인하고 작업해야합니다.



그러나 **확인하지 못해 동일한 작업을  
중복으로 수행하게 되는 상황이**  
자주 벌어지고는 합니다.

“

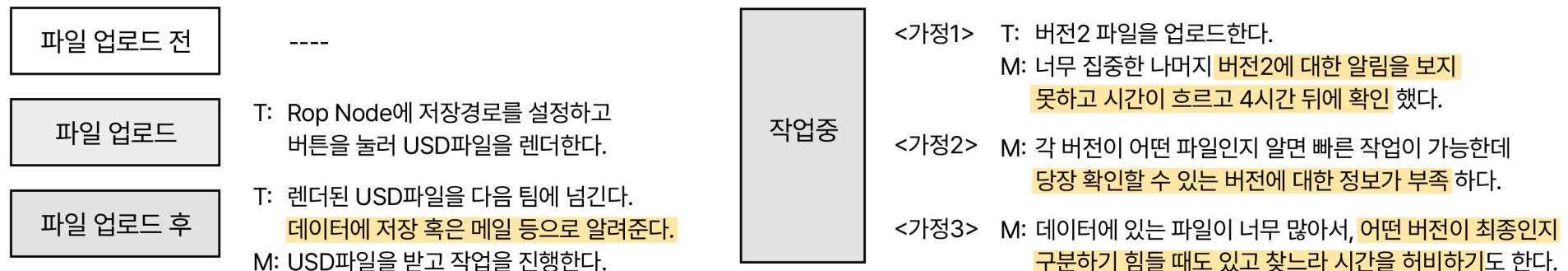
버전 업로드 알림이 없다는 것만으로 벌어지는 문제가 맞을까?  
**표면적인 문제 말고, 근본적인 문제가 무엇일까?**

”

## 문제 발견 | issue finding

<알림을 주는 것이 근본적인 해결책이 맞는지>

<알림이 없어도, 버전업 이후 작업에 부정적 영향이 끼치지 않을 해결책은 없는지>

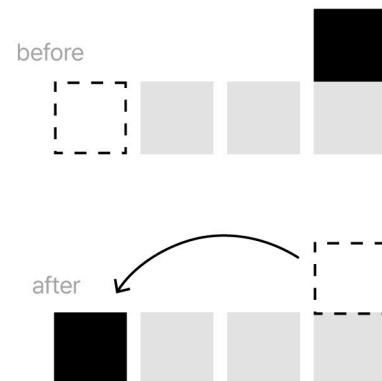


아티스트가 작업을 하고, 업로드하는 과정의 순서를 크게 4가지로 나눠보고, 각 상황 속에서 어떤 문제를 겪을 수 있을지 파악하고자 했습니다.

### 경험할 수 있는 사소하지만 불편한 문제들

1. 파일을 업로드한 후, 어떤식으로 파일을 새로 업로드했다고 전달해야 할지, 누구에게 전달해야 할지 매번 번거로울 수 있음  
또는 버전업 파일 알림을 따로 주지 않고, 회사내부 스토리나 클라우드에 저장을 했을 때 그 파일을 보지 못할 때 생기는 문제
2. 매번 연락이 올 수 있는 상황을 기다릴 수가 없음. 메일이나 톡같은 경우 대화들이 섞여 중요한 알림은 구분하기 어려울 수 있음
3. 수정사항이든, 중요한 정보를 아티스트가 따로 기입을 하거나 저장을 해야하는 번거로울 수 있음
4. 바쁜 나머지 파일을 저장할 때 파일명을 버전업 해서 저장하지 못하는 경우, 버전이 덮어 씌워지는 문제가 생길 수 있음
5. 어떤 버전이 최종인지 구분하기 어렵고 찾느라 시간을 낭비하는 경우가 있음
6. 매번 파일을 저장할 때, 어떤 씬, 어떤 시퀀스에 포함되어있는 파일인지 헷갈릴 때가 있을 수 있음
7. 다양한 버전이 있을 때, 특정 버전을 확인하고 싶은 상황이 생길 수 있음. 하나씩 노드에 연결해서  
열람을 하며 확인해야 하는데 반복되는 작업이라 시간을 낭비할 수 있음
8. 어떤 작업자가 파일을 작업했는지 궁금할 수 있는데, 알아보기까지의 과정이 까다로울 수 있음
9. 버전이 많은 경우, 어떤 파일이 최종 파일인지 소통의 오류로 고칠 수도 있음
10. 버전에 대한 선택과 사용이 빠르게 이루어지지 못하고 시간이 걸려서 효율이 떨어지는 문제가 있음

## 문제 재정의 및 구체화 | idea & thinking



파일이 버전업이 되는 과정 중, '작업중'에 사용자의 경험이 몰려있고, 대부분의 경험은 페인포인트 ...  
'파일 업로드 전' 단계에서는 작업을 하고 저장하고 업로드하는 것 외에는 특별한 경험은 없어보였습니다.

일의 효율성과 능률을 위해 아티스트가 작업을 하는 동안엔 최소한의 방해만 받을 수 있도록,  
'작업중'일 때 일어나는 일들이 '파일 업로드 전' 단계에서 이루어져야 일의 분배가 알맞겠다고 생각하였습니다.

그래서, 앞팀이 파일 업로드 전 많은 정보를 주고, 그 정보를 뒷팀에서 긍정적인 방향으로 사용하는 것을 문제 해결의 방향으로 설정하였습니다.



작업자가 빠르게 정보를  
확인할 수 있도록 UI를 활용해  
직관적인 형태로 보여주기



최종 파일 하나를 지정해두고  
그에 따른 버전들을 편리하게 확인하고  
접근할 수 있는 환경 제공하기



메일/사내알림/메신저 등  
다양한 정보가 가득한 곳과  
분리된 버전알림 제공하기

# 문제 해결 | problem solving

## 1. 버전 업로드



초반에는 렌더 버튼 하나를 만들어서, usd\_rop 노드가 있다면 그 노드를 렌더하는 방식으로 코드를 작성했습니다. 그러나, **usd\_rop 노드가 여러개가 있을 경우엔 한꺼번에 모든 노드를 읽어서 렌더하는 과정 및 db업로드에 대한 오류**가 있었습니다. 그래서 usd rop 노드 각각에 커스텀 렌더 버튼을 부여하기 위해서 hou.ButtonParmTemplate() 메서드를 사용했습니다.

```
1 fin_dir = pathlib.Path(final_path) / fin_name
2 if not fin_dir.exists():
3     os.symlink(src_dir.as_posix(), fin_dir.as_posix())
4     print('link created succ')
5 else:
6     print('succ')
```

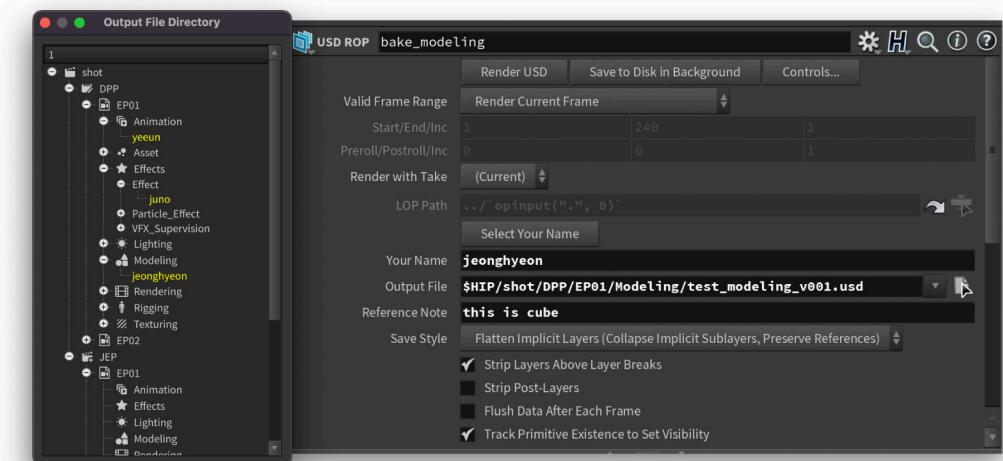
usd\_rop\_OnCreated.py

ButtonParmTemplate()로 Render USD라는 커스텀 버튼을 생성했고 버튼을 누르면 **파일 경로, 작업자 이름, 노트를 읽어서 db에 저장하고 파일과 저장되는 현재 버전 파일이 심볼릭 링크로 연결**이 됩니다.

Select Your Name: 이름과 팀에 맞게 아웃풋 경로 및 이름이 설정됨

Your Name: 직접 입력 또는 이름 선택

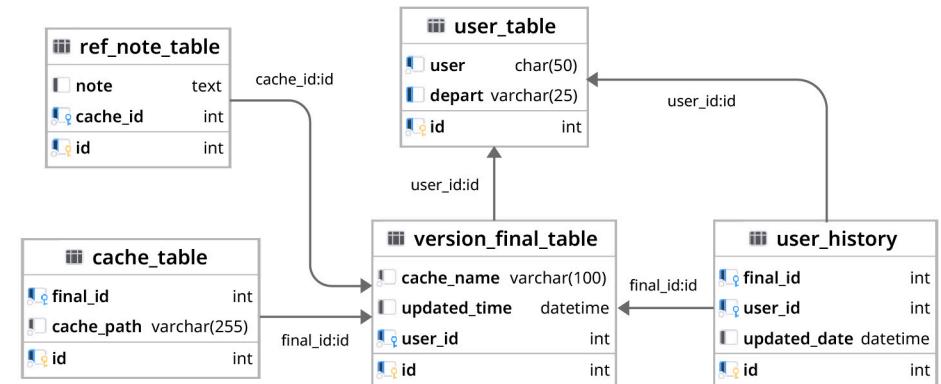
Reference Note: 필요한 정보 및 수정사항에 관한 내용 기입 가능



USD ROP 노드 및 Select Your Name을 누르면 볼 수 있는 모습



0| 부분에서 심볼릭링크가 적용이 됨 (렌더 이후에 벌어질 상황)



version up DB Schema

## 2. 버전 알림

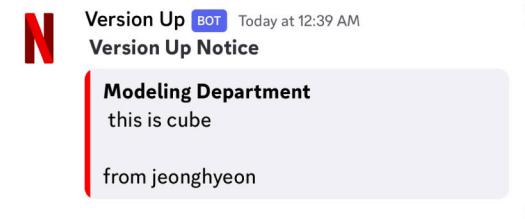


버전 업로드에서 자동 및 직접 기입했던 정보(파일 경로, 작업자 이름, 노트, 업로드 된 날짜)가 db에 저장이 되는 동시에, **디스코드 API를 활용해 버전업이 될 때만 알림을 받을 수 있는 분리된 공간**을 만들었습니다.

```
def slot_msg():
    try:
        note = hou.parm('refnote').eval()
        user = hou.parm('username').eval()
        depart = f''' {__db.get_depart_by_name(user)[0]}
[0]} Department '''
        msg = f'{depart}\n {note}\n\n from {user}'
        set_msg(msg, 16711680)
    except AttributeError:
        return
```

db\_up.py

디스코드 알림 모습



## 3. 버전 피팅



db의 `version_final_table`에 저장된 파일 파일 개수만큼 위젯을 만듭니다. 파일 파일 아이디를 가진 `cache_file`을 추출 → combobox 아이템 지정 및 `cache file` 관련된 모든 정보를 info table view의 리스트로 생성했습니다. `reload`는 콤보박스 index값을 및 file path로 변환하여 다시 `symlink`를 걸어주고, 이는 사용자가 자유롭게 버전 피팅이 가능하도록 해주는 중요 역할을 합니다. MVC로 table view를 시각화했고 연결된 버전에 role로 컬러값을 주었습니다.

	id	cache_name	updated_time	user_id
1	1	test_modeling.usd	2024-03-25 00:41:42	3
2	2	test_rigging.usd	2024-03-25 00:43:35	9

	USER	DEPART	CACHE_NAME	NOTE	UPDATED_TIME
1	jeonghyeon	Modeling	/Users/yeko/shot/DPP/EP01/Modeling/test_modeling_v001.usd	this is cube	2024-3-25 0:39
2	jeonghyeon	Modeling	/Users/yeko/shot/DPP/EP01/Modeling/test_modeling_v002.usd	this is rubbertoy	2024-3-25 0:40
3	jeonghyeon	Modeling	/Users/yeko/shot/DPP/EP01/Modeling/test_modeling_v003.usd	this is shaderball	2024-3-25 0:41

```
if role == QtCore.Qt.BackgroundRole:
    if index.row() == self.symlink:
        return QtGui.QColor('green')

def setup_widget(self):
    f_name = self.__db.get_final_name()
    f_id = self.__db.get_final_id()
    for i, n in enumerate(f_name):
```

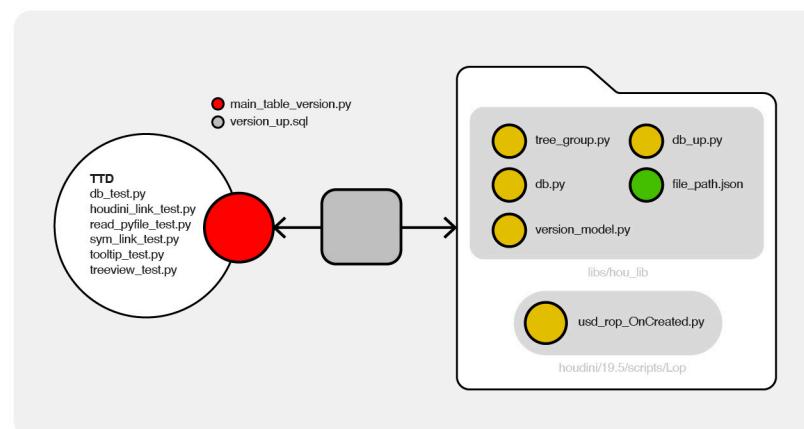
main\_table\_version.py

# 최종 결과 및 시연 | final result

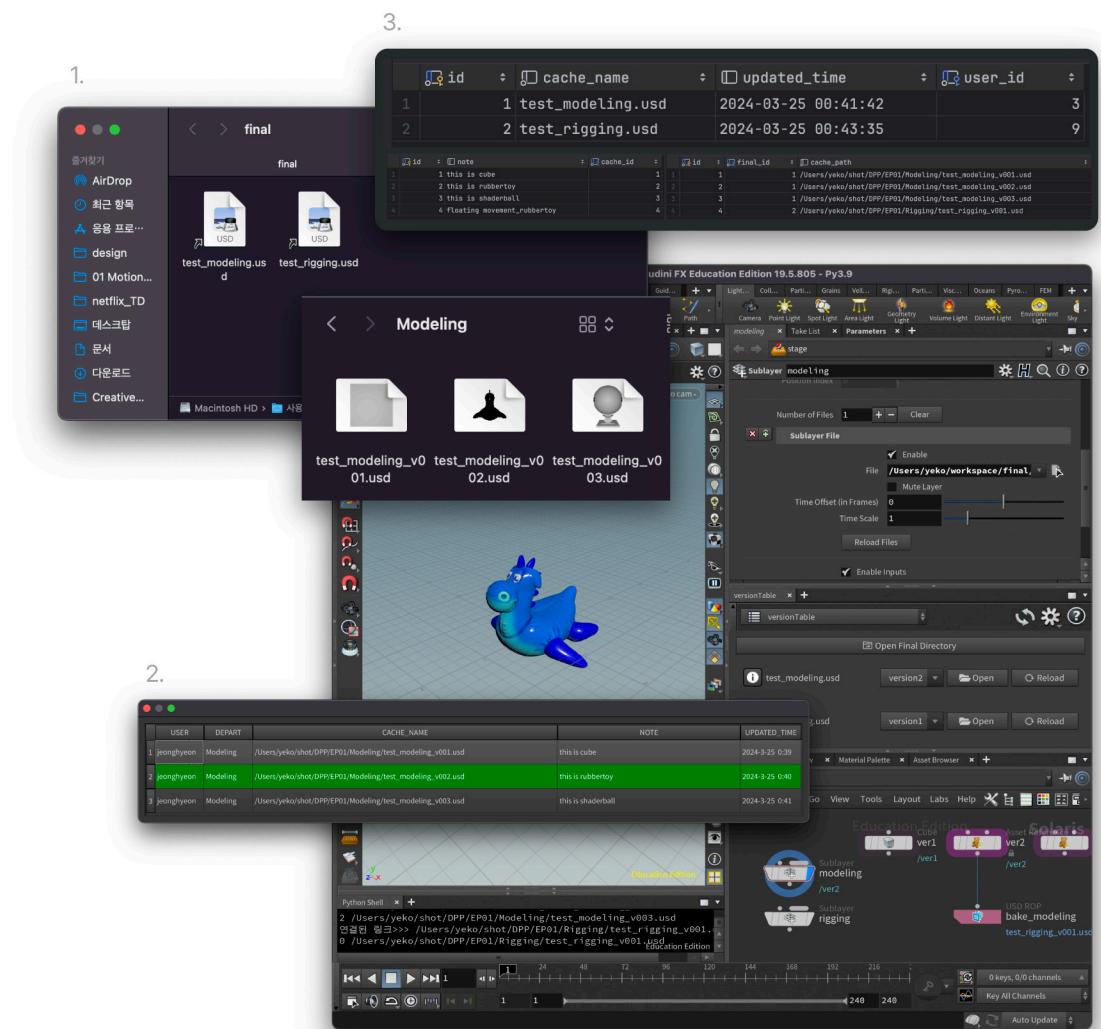
## 1. 미리보기 (Preview)

모델링 파일이 버전 3개가 있고, 테이블뷰를 통해서 현재 버전 2로 연결이 되어있습니다.  
또한, 현재 sublayer를 통해 final file을 열고 있는데, 버전 2파일로 잘 열리고 있음을 볼 수 있습니다.

## 2. 코드 의존성 (code dependency)

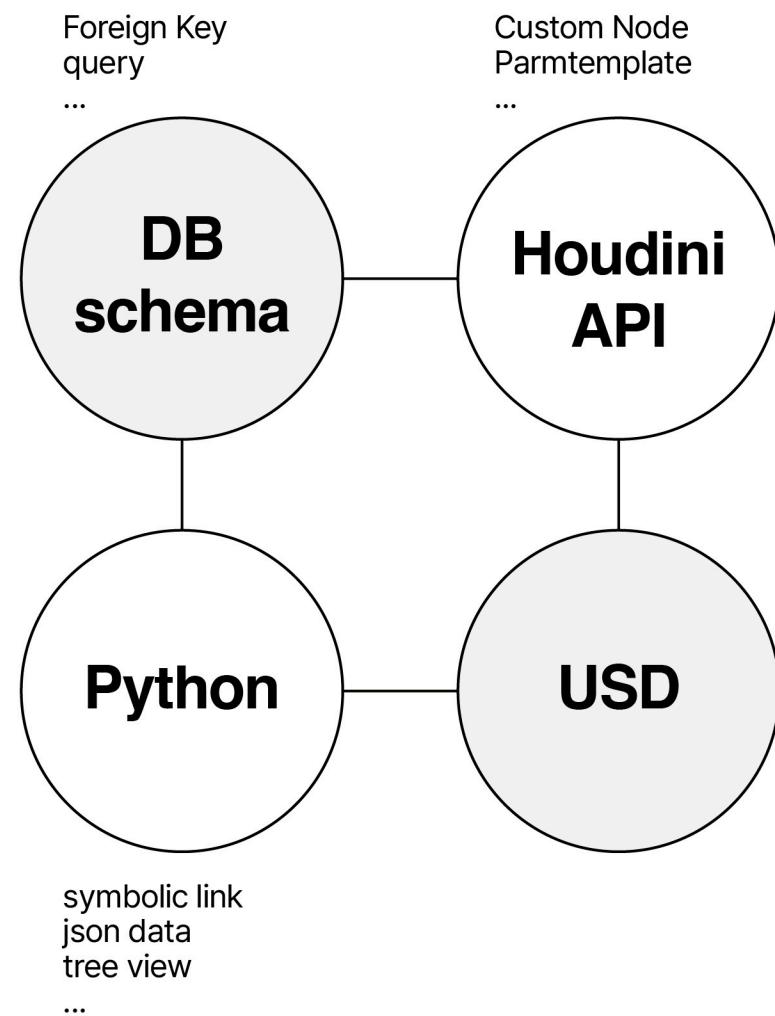


메인 테이블 버전 파일과 버전업 데이터베이스(sql)을 중심으로 코드의 의존성이 이루어지고 있습니다.



- 파이널 파일이 따로 있고 캐시 파일이 따로 있는 모습
- DB의 데이터를 가지고 와서 버전들의 정보를 보여주고 있는 모습
- DB의 모습

## 배운점 | what I learn



db schema를 짜는 과정부터, db를 활용하는 것.  
후디니에 내장된 노드 자체를 숨기고 커스텀 노드를 적용시키는 것.  
트리뷰와 테이블뷰, 제이슨 데이터와 db 데이터.  
그 외에 아티스트가 겪을 수 있는 문제점들을 파악해보는 것까지.

개인 프로젝트를 통해서, 지금까지 공부해온 파이썬 코드들을 활용하며  
개념도 다시 잡아볼 수 있었고, 코드 외에도 개발 구성, 코드 기획 등을  
어떤 방향으로 설계하고 진행하는지 조금은 알 수 있었던 것 같습니다.  
특히 오류를 최소화하기 위해서 테스트했던 코드들이 메인 코드를  
효율적으로 짜도록 도와준다는 걸 배웠습니다.

넷플릭스 VFX 아카데미 TD 개인 프로젝트 발표 Version Pinning

# Thank you

3기 교육생 고예은