KossLab
KOREA OPEN SOURCE SOFTWARE DEVELOPERS LAB
공개SW개발자Lab

HTML5 기반의 데이터베이스 사용자 도구

Planche

공개SW개발자Lab 오픈소스프론티어 2기 정주원

KossLab

개발자들이 데이터를 쉽게 보기 위해 사용하는 도구를 "데이터베이스 클라이언트 도구" 라 부른다. 기존의 수많은 설치형 애플리케이션이 존재한다. 그러나 지금은 바야흐로 웹이 주도하는 시대이다. 구글 드라이브 같은 문서 도구를 예를 들자면 그동안 설치해서 사용했던 워드나, 엑셀, 파워포인트 같은 제품들이 웹 기반의 기술로 만들어져 브라우저를 통해 사용자에게 서비스 되는 것이다.

이제부터 설명하고자 하는 플란체(Planche) 프로젝트는 새로운 MySQL 데이터베이스의 도구에 대한 것이다. 프로젝트에 대한 소개와 HTTP 터널링 그리고 쿼리 토큰화 과정에 대한 내용을 다룬다.

[목차]

1. 데이터베이스 사용자 도구

데이터가 기하급수적으로 증가하는 시대이다. 온라인으로 쇼핑을 하며 물건을 구매할 때나 어느 특정한 사이트에 회원 가입을 하도 혹은 카톡 같은 메신저를 사용할 때도 데이터는 계속 쌓이게 마련이다.

무수한 데이터는 사용자들이 볼 수 있는 정보로 가공이 필요하다. 이를 위해 개발자들은 서비스 개발 단계에서 어딘가에 쌓여 있는 데이터들을 의미 있는 형태의 데이터 집합으로 묶고 가공을 거친 후 최종적으로 사용자의 서비스 화면에 표현된다.

자연스럽게 개발 단계에서 쌓여 있는 데이터를 개발자들이 쉽게 다루기 위해서 데이터베이스 클라이언트 도구를 찾게 된다. 다행히 이미 세상엔 수많은 회사와 개발자들이 만든 여러 클라이언트 도구들이 존재한다.

이제부터 이야기할 도구는 MySQL 데이터베이스의 도구에 대한 것이다. MySQL 진영에도 여러 편리한 도구들이 있다. 여러분들이 쓰는 많은 윈도우 프로그램들처럼 그 동안 데이터베이스 클라이언트 도구 또한 사용자의 PC 에 설치돼 사용하고 있다.

그러나 지금은 바야흐로 웹이 주도하는 시대이다. 구글 드라이브 같은 문서 도구를 예를 들자면 그 동안 설치해서 사용했던 워드나 엑셀, 파워포인트 같은 제품들이 이미 수년 전부터 웹 기반의 기술로 만들어져 사용자에게 서비스되고 있다.

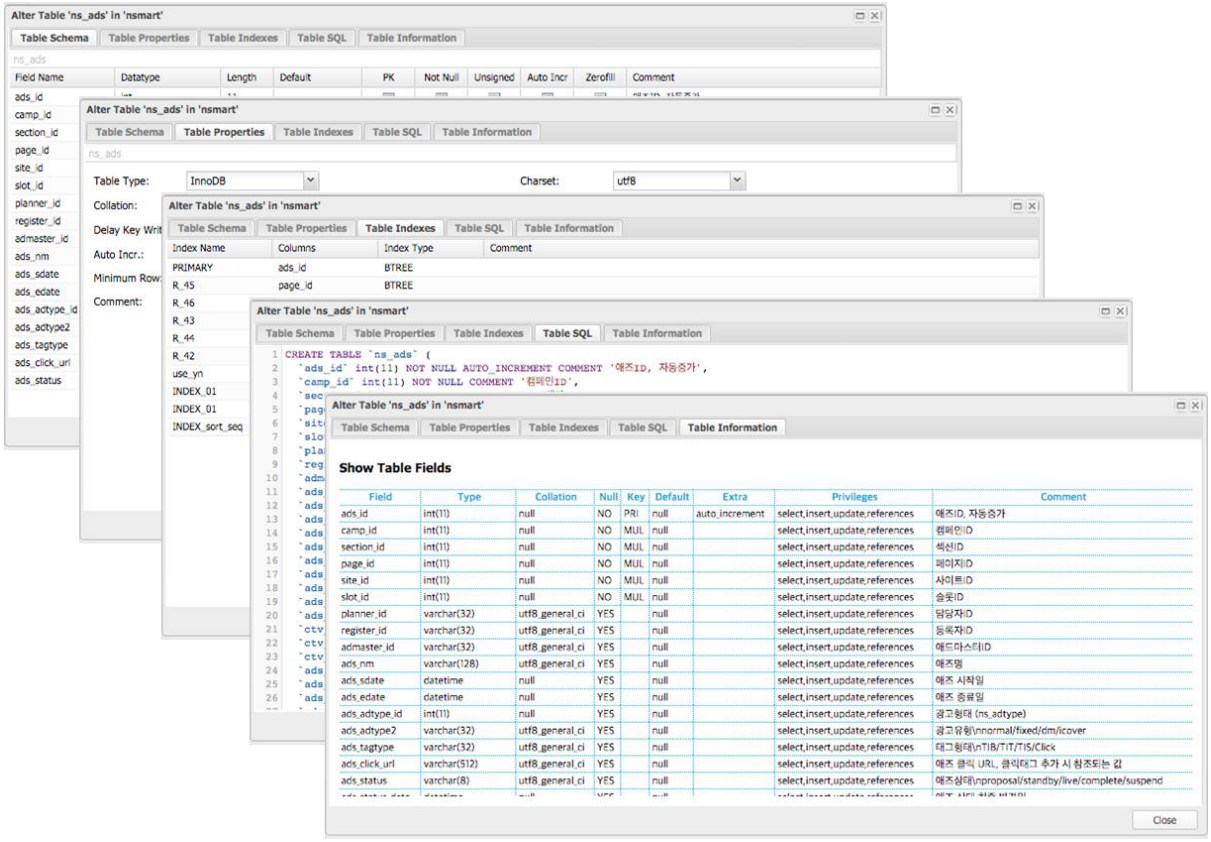
[그림 1] 구글 드라이브

"웹 기반의 데이터 베이스 사용자 도구 플란체(Planche)" 또한 MySQL 클라이언트 도구이며 웹 기술을 이용하여 브라우저를 통해 사용할 수 있다.

2. 브라우저에서 실행되는 도구

MySQL 진영에는 오래 전부터 웹 기술을 기반으로 서버에 설치해 사용하는 클라이언트 도구가 존재한다. 그러나 개발자들도 설치형 도구가 제공하는 간편한 설치와 실행에 익숙하다. 많은 개발자들이 유료 설치형 도구를 개발할 때 사용하고 있다.

플란체는 웹 기술로 만들어져 있어 다운로드 받고 압축을 해제하고 별도의 설치 없이 실행이 가능하다. 또한 서버에 설치할 경우 다수의 사용자가 공유할 수 있는 멋진 도구로 변하는 것이 특징이다.



[그림 2] 플란체의 스키마 탭

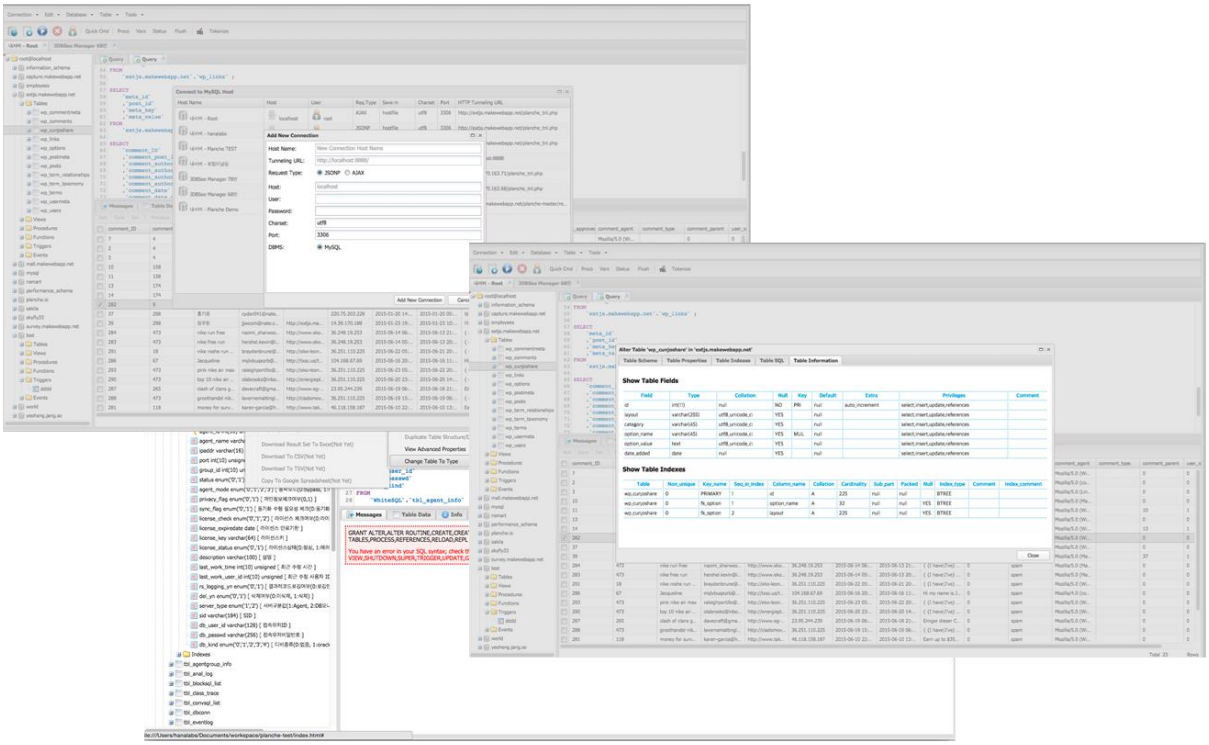
플란체는 유료의 설치형 MySQL 클라이언트 도구가 기본적으로 웹에서 제공하는 기능을 비슷하게 구현하자는 목표를 담았다. 웹 상에서 설치형 클라이언트 도구의 모습을 흉내내기 위해 Sencha ExtJS Framework 를 활용했다.

[그림 3] 대표적인 다섯 종류의 브라우저

웹으로 구현하며 어떤 OS 든 브라우저만 설치되어 있다면 동작할 수 있는 환경을 제공한다. 대표적인 다섯 종류의 브라우저(인터넷 익스플로러, 파이어 폭스, 크롬, 오페라, 사파리) 에서 사용이 가능하다. DBA(데이터베이스를 다루는 기술자)나 개발자들이 가볍고 빠르게 사용할 수 있는 환경을 제공하고 아울러 유익한 플러그인들을 개발할 수 있도록 유연한 구조를 제공한다.

3. 플란체 소개

플란체는 MySQL GUI 클라이언트 도구이다. 브라우저를 통해 실행이 가능하며 HTTP 터널링 방식을 사용하여 데이터베이스 서버의 데이터를 컨트롤하고 쿼리할 수 있다. Github 를 통해서 다운로드 받을 수 있으며 웹의 유연한 빌드 방식 덕에 여러 버전을 현재 개발 단계에서도 제공하고 있다.



[그림 4] 플란체의 실행화면

(1) orginial version

플란체의 기본 버전으로 서버나 사용자의 개인 개발 환경에 설치하여 사용할 수 있다.

Github : https://github.com/plancheproject/planche

(2) wordpress version(wordpress plugin)

워드프레스 플러그인 버전으로 개발돼 클릭 몇 번으로 워드프레스의 관리자 화면에서 관리자가 별도의 설정 없이 접속할 수 있다. 기본적으로는 워드프레스를 이용하여 터널링하기 때문에 별도의 터널링 서버가 필요 없다.

Github : https://github.com/plancheproject/planche-wp

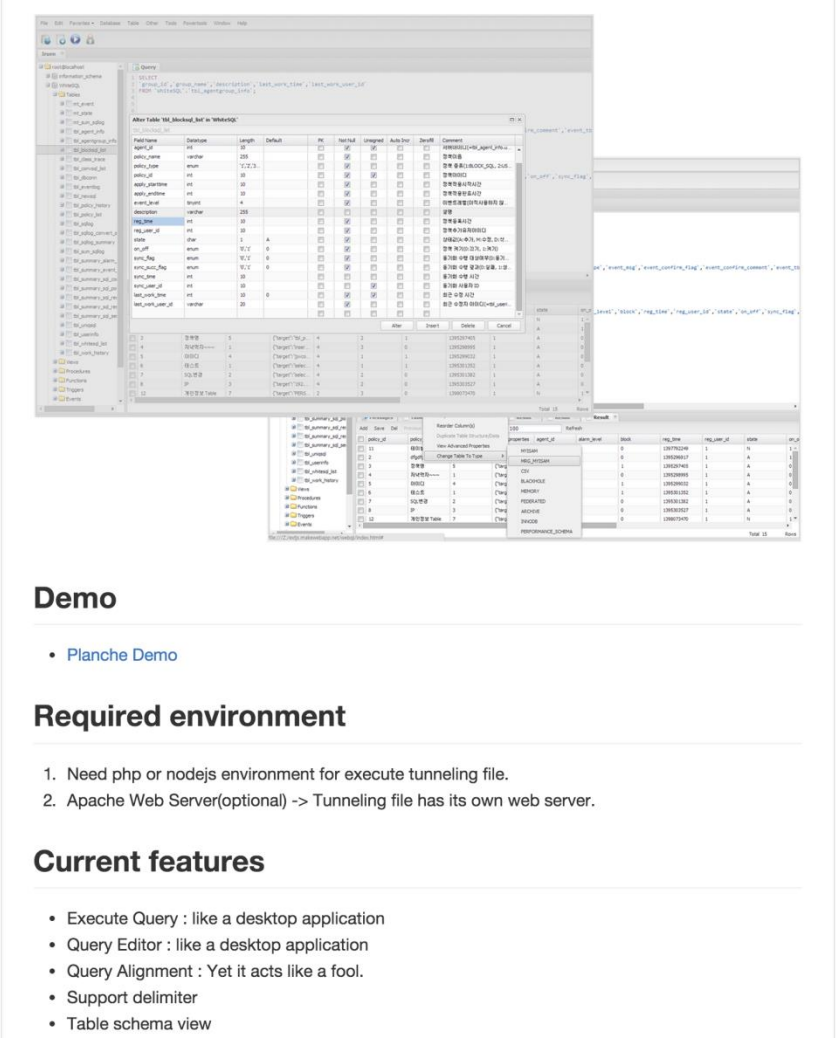
(3) desktop version(electron 패키징)

일렉트론(Electron)을 통해 윈도우 또는 맥 등에서 설치하여 사용할 수 있도록 패키징 되어 개발되고 있는 버전이다. 패키징이 되므로 터널링 서버를 앱 구동 시 내부에서 별도의 프로세스를 띄워 실행하는 방식으로 구현한다.

[그림 5] 데스크탑 패키징을 도와주는 일렉트론

Github : https://github.com/plancheproject/planche-desktop

Electron : http://electron.atom.io



Demo

- Planche Demo

Required environment

1. Need php or nodejs environment for execute tunneling file.
2. Apache Web Server(optional) -> Tunneling file has its own web server.

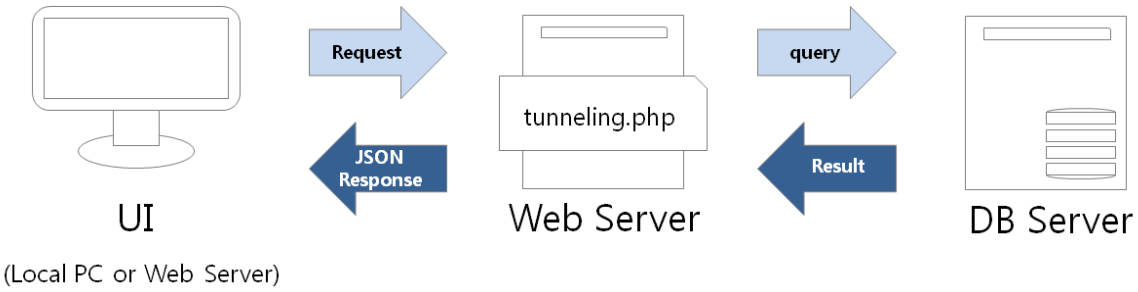
Current features

- Execute Query : like a desktop application
- Query Editor : like a desktop application
- Query Alignment : Yet it acts like a fool.
- Support delimiter
- Table schema view

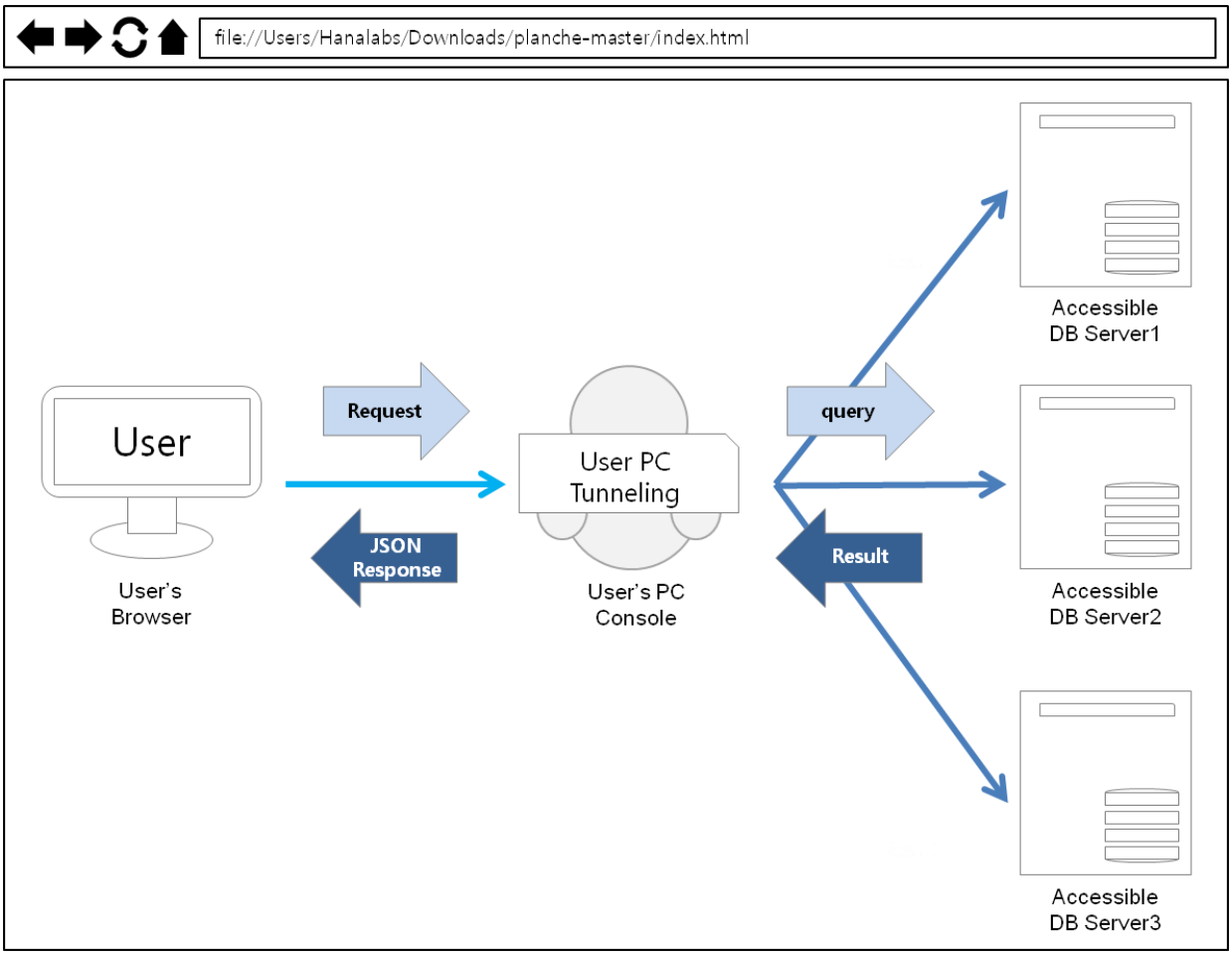
[그림 6] GitHub - Planche

4. HTTP 터널링(Tunneling)

플란체는 자바스크립트 기반이므로 데이터베이스에 접근할 수 있는 별도의 Client API 가 존재하지 않는다. 그래서 서버사이드 언어를 이용한 터널링 방식을 사용한다. 터널링 소스코드는 매우 간단하며 사용자 UI 에서 요청한 명령을 받아 실행하는 역할만 한다. 그래서 전체 애플리케이션 소스코드의 양 대비 매우 적은 양으로 존재한다.



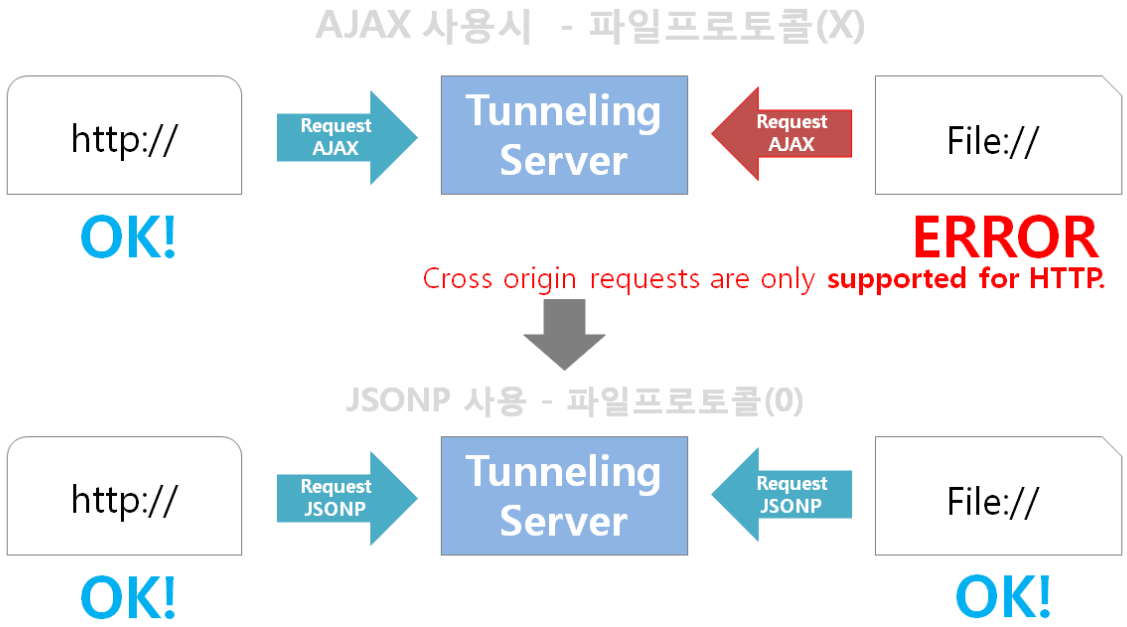
[그림 7] 웹서버를 통한 HTTP 터널링 방법



[그림 8] 로컬 터널링을 통한 HTTP 터널링 방법

터널링 파일만 올라가 있다면 내부(로컬) 접속이 가능한 데이터베이스도 웹 서버에 설치된 터널링 파일을 통해 우회적인 접근이 가능하다. 터널링 기술은 클라이언트와 서버간의 내부적으로 약속된 통로를 만들어 놓고 이를 통해 데이터를 주고 받는 방법이다.

Ajax 데이터 통신기술과 JSOP 라는 방법을 선택적으로 사용하여 서버에 설치해 사용하거나 사용자의 PC 에서 바로 실행하더라도 구동되도록 JSONP 통신방식을 채택하여 크로스 도메인 이슈에서도 벗어 날 수 있다.



[그림 9] 크로스 도메인 이슈를 피할 수 있는 JSONP

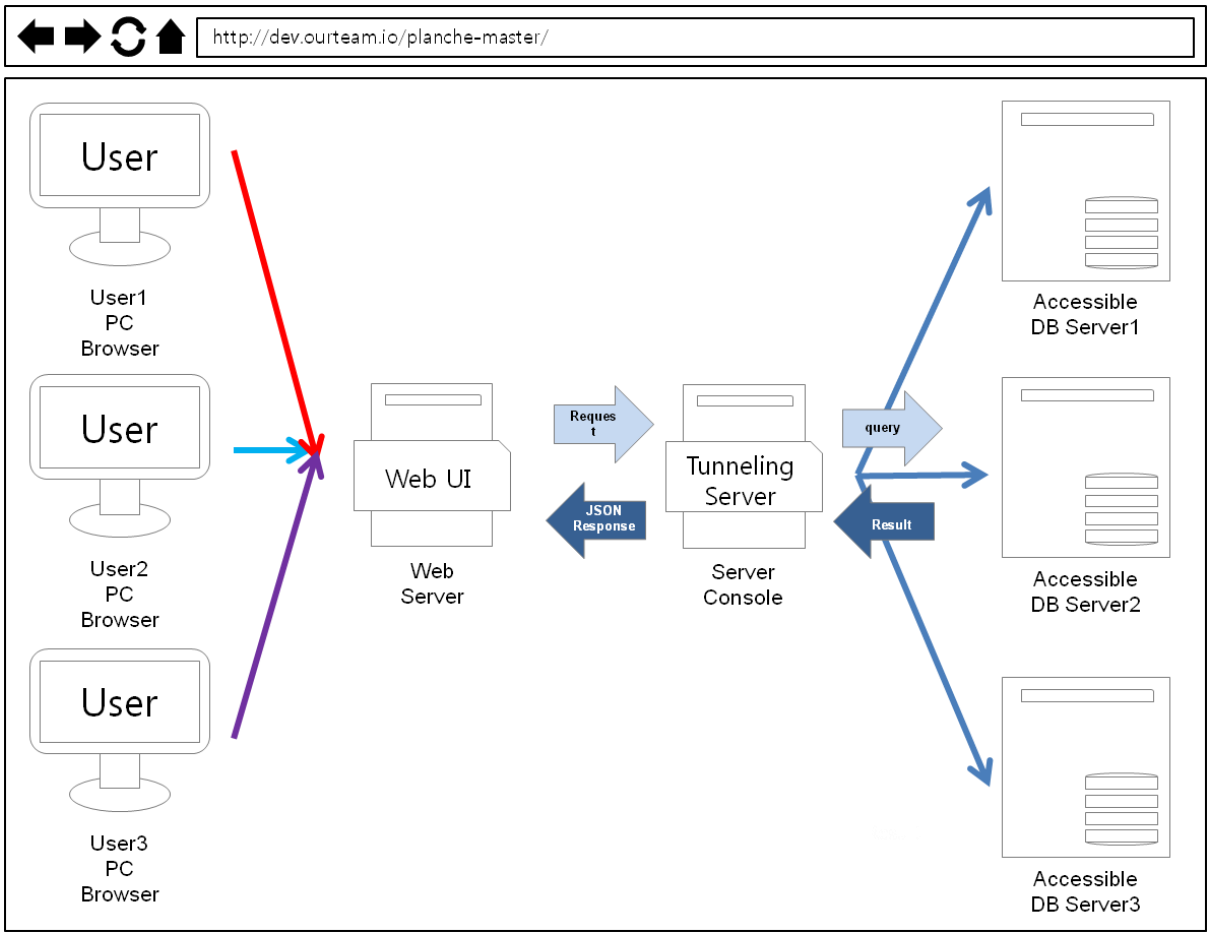
#크로스 도메인 이슈(Cross Domain Issue) 간단 설명

웹 문서 내에서 자바스크립트 혹은 AJAX 를 통해 서로 다른 서버에 통신을 할 때 걸리는 보안 제약

5. 팀 내에 도구로 사용

플란체의 장점은 웹서버에 올리고 실행하는 경우 각 개별 팀원들이 별도의 설치 없이 팀 도구로 사용할 수 있는 장점이 있다. 팀 내에 공유 서버 등에 플란체를 설치하고 관리자는 호스트 접속

정보 등을 미리 셋팅하여 팀 메일이나 메신저로 배포할 수 있다. 팀원들은 별도의 접속 정보 설정 없이도 바로 관리자가 미리 셋팅해 놓은 정보를 통해 사용이 가능하다.

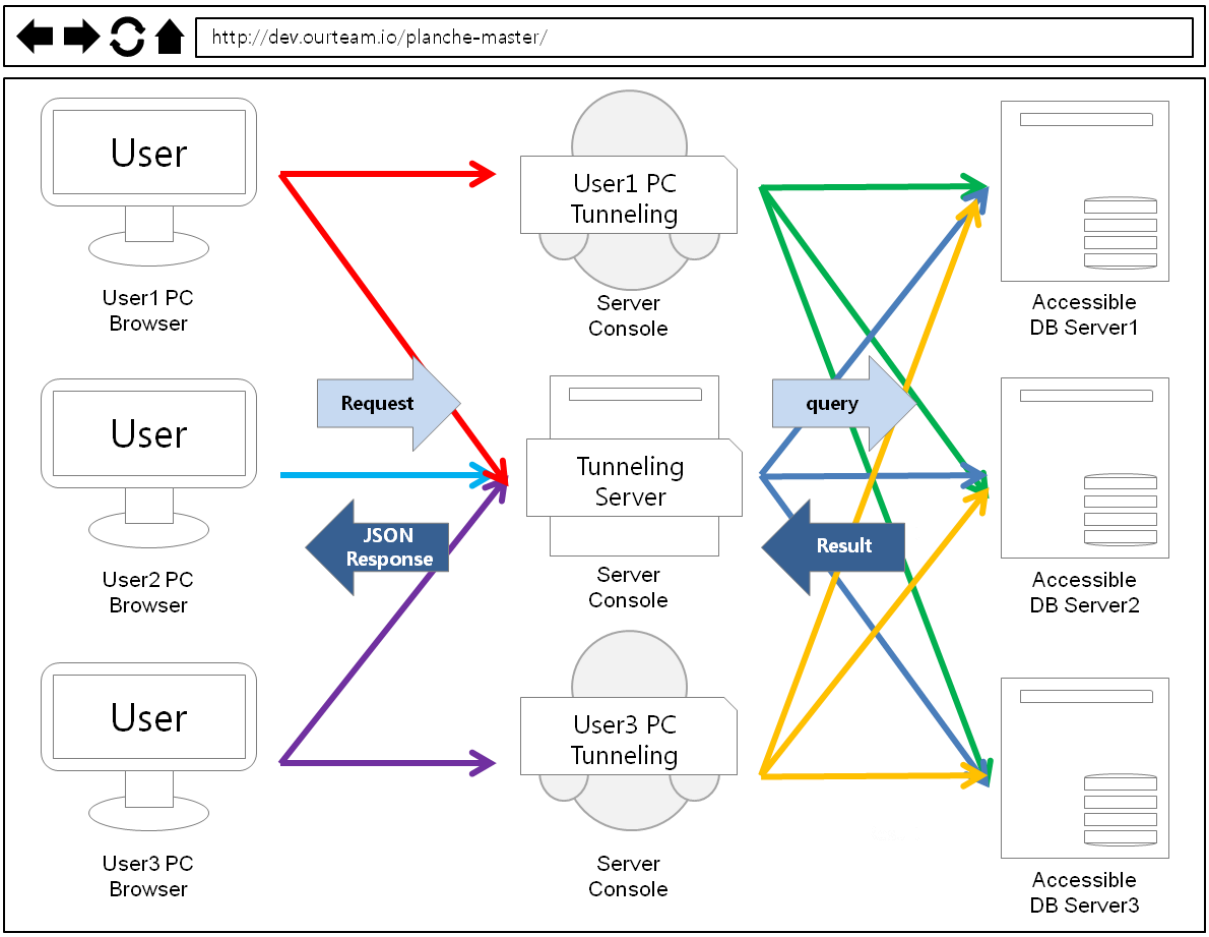


[그림 10] 플란체를 다수의 사용자가 공유하는 모습

6. HTTP 터널링 다중 환경 구성 (Tunneling)

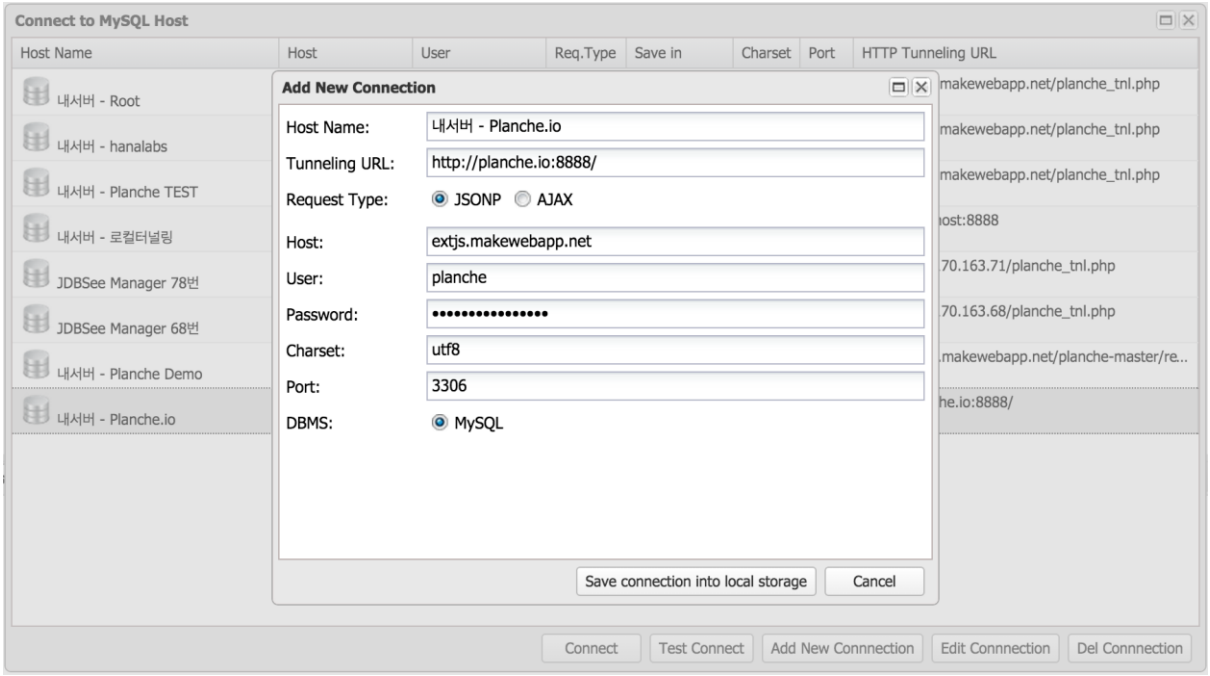
터널링 서버는 사용자의 개인 개발 환경뿐 만 아니라 다수가 사용하는 공유 서버에서도 설치할 수 있다. 공유 서버에 설치하게 되는 경우 사용자는 해당 공유 서버의 터널링 환경을 이용하게 되며 별도로 각 사용자가 터널링 환경을 구축 할 필요가 없다.

각 데이터베이스(DB Server)들은 접속이 허용된 웹 서버에서 터널링 로직이 요청한 쿼리를 실행하고 결과를 사용자에게 전송한다.



[그림 11] 여러 터널링 서버를 두고 다수의 사용자가 공유하는 모습

유저는 단순히 플란체가 실행된 상태에서 접속 정보 추가창을 실행하여 터널링 주소와 기본적으로 통신할 데이터베이스의 접속 정보를 입력하면 된다. 접속창에선 JSONP 를 지원하므로 터널링 주소가 현재 플란체가 설치되어 실행되는 도메인과 다르다고 해도 별다른 문제없이 통신할 수 있다.



[그림 12] 커넥션 추가시 터널링 URL 을 포함한 정보를 저장

7. SQL 쿼리 분리

플란체에서 실행되는 쿼리들은 모두 에디터에 입력이 되고 실행이 된다. 사용자는 하나의 에디터 영역에 하나의 쿼리만 입력하고 실행하지 않는다. 여러 쿼리가 입력된 상황에서 어떻게 쿼리를 분리해서 실행할지는 전적으로 프로그램 내의 로직을 통해서 분리를 해줘야 한다.

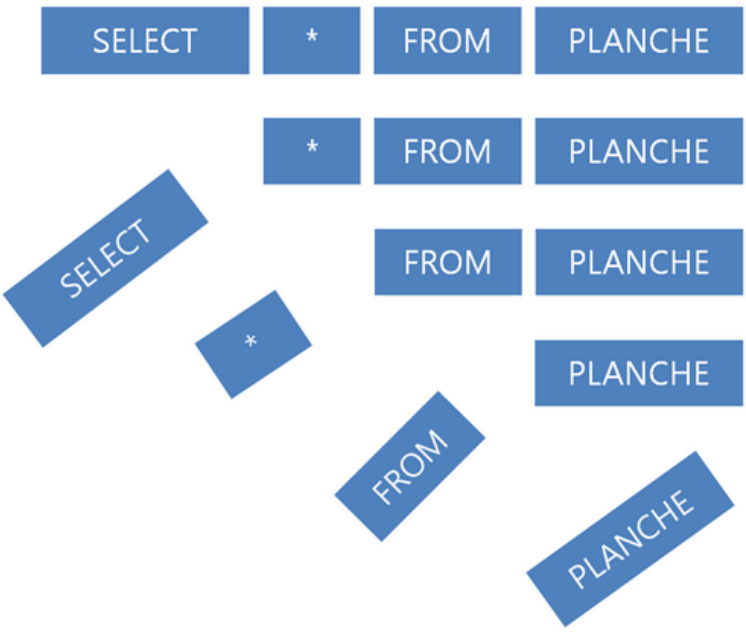
```
1  DELIMITER $$
2
3  USE `test_db_1`$$
4
5  DROP PROCEDURE IF EXISTS `proc3`$$
6
7  CREATE DEFINER=`websql`@`localhost` PROCEDURE `proc3`()
8  BEGIN
9
10   SELECT`uniq_id`,`camp_id`,`freq`,`uid`
11   FROM `test_db_1`.`rs_connect_camp`;
12
13 END$$
14
15 DELIMITER;
16
17 SELECT 1;
18
```

[그림 13] MySQL 의 Delimiter

```
1  DELIMITER $$
2
3  USE `test_db_1`$$
4
5  DROP PROCEDURE IF EXISTS `proc3`$$
6
7  CREATE DEFINER=`websql`@`localhost` PROCEDURE `proc3`()
8  BEGIN
9
10   SELECT`uniq_id`,`camp_id`,`freq`,`uid`
11   FROM `test_db_1`.`rs_connect_camp`;
12
13 END$$
14
15 DELIMITER;
16
17 SELECT 1;
18
```

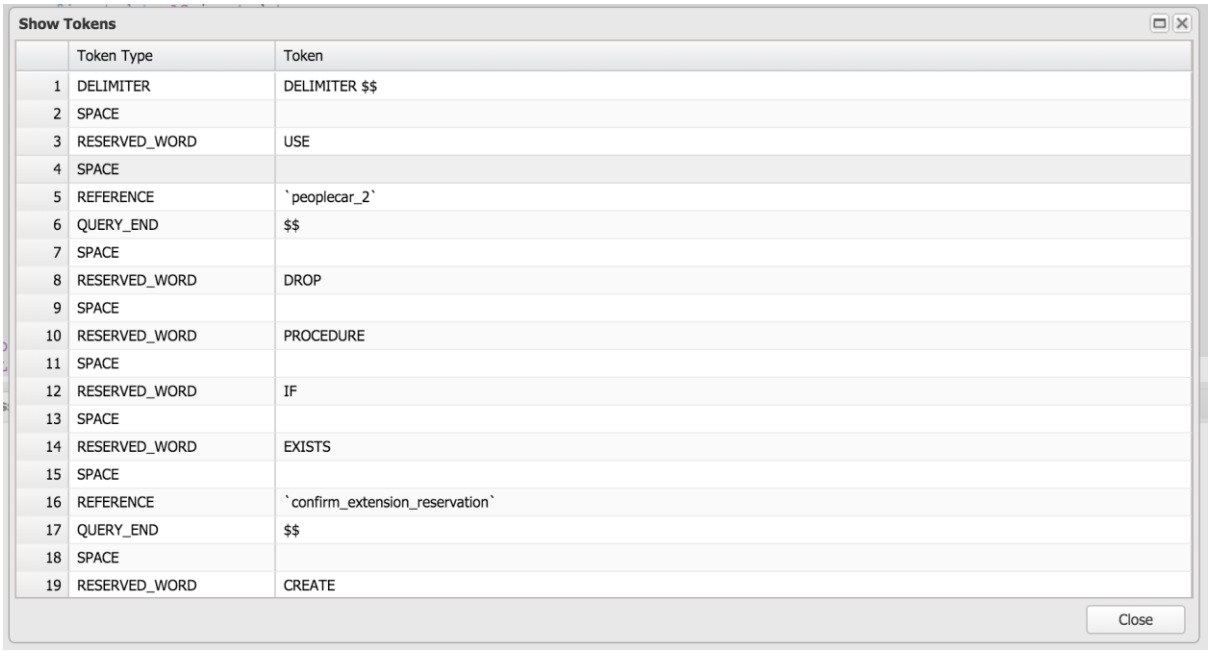
[그림 14] 변경된 Delimiter 에 의한 분리

입력된 다수의 쿼리를 분리하는 작업은 비교적 단순하다. 일단 입력된 쿼리를 어절 단위로 분리한다. 그리고 이미 구축해 놓은 키워드 또는 예약어 목록과 매칭해 보고, 분리되어 들어온 문자열이 구축해 놓은 키워드 목록에서 어떤 키워드와 일치하는지를 판단하여 해당 키워드가 갖게 될 특성을 설정하는데 이를 '토큰화' 한다고 표현한다.



[그림 15] 쿼리의 어절 단위 분리 - "SELECT * FROM PLANCHE"

토큰화를 하는 과정의 내부를 들여다 보면 앞에서 말한대로 문자열을 띄어쓰기 기준인 어절 단위로 잘라내는 전처리 과정을 거친다. MySQL 에는 Delimiter 라는 사용자가 언제든 지정 할 수 있는 쿼리 종료 문자열이 있다. 기본 설정은 ";" 세미콜론을 따르는데 저장 프로시져나 함수 등의 쿼리를 작성할 때는 전략적으로 "$$"등으로 Delimiter 를 바꾸어서 작성을 하게 된다. 토큰화하고 난 후 Delimeter 가 ";"인지 "$$"인지 아니면 기타 다른 무엇인지 판단이 되면 해당 Delimiter 를 기준으로 쿼리를 분리해 낼 수가 있기 때문에 전처리 과정은 꼭 필요한 단계이다.

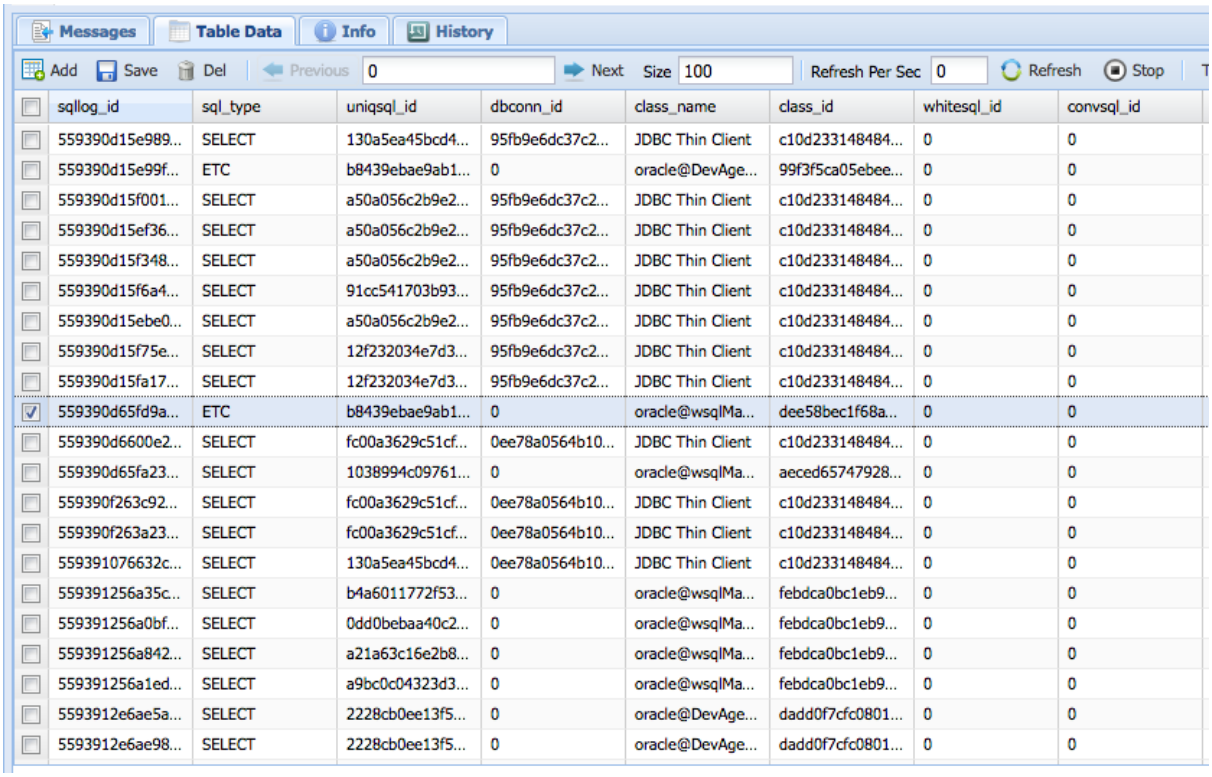

Show Tokens

	Token Type	Token
1	DELIMITER	DELIMITER $$
2	SPACE	
3	RESERVED_WORD	USE
4	SPACE	
5	REFERENCE	`peoplecar_2`
6	QUERY_END	$$
7	SPACE	
8	RESERVED_WORD	DROP
9	SPACE	
10	RESERVED_WORD	PROCEDURE
11	SPACE	
12	RESERVED_WORD	IF
13	SPACE	
14	RESERVED_WORD	EXISTS
15	SPACE	
16	REFERENCE	`confirm_extension_reservation`
17	QUERY_END	$$
18	SPACE	
19	RESERVED_WORD	CREATE

Close

[그림 16] 토큰화한 쿼리의 모습

이렇게 분리된 쿼리는 한번에 한 개씩만 터널링 서버를 통해 실행되어 그 결과를 JSON 데이터로 돌려주게 되고 플란체는 이를 읽어 데이터를 비로소 아래와 같은 시각적 데이터 테이블 형태로 표시하게 된다.

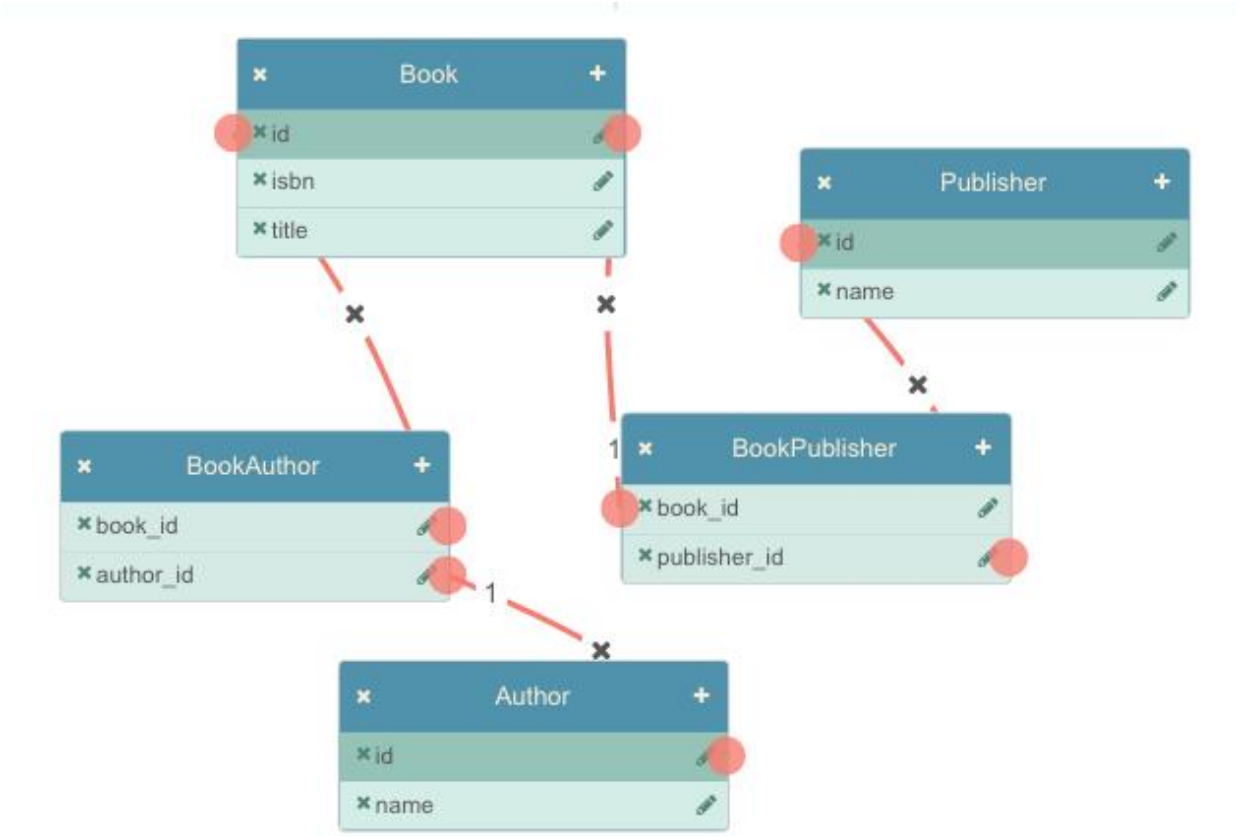


sqllog_id	sql_type	uniqsql_id	dbconn_id	class_name	class_id	whitesql_id	convsql_id
559390d15e989...	SELECT	130a5ea45bcd4...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d15e99f...	ETC	b8439ebae9ab1...	0	oracle@DevAge...	99f3f5ca05ebee...	0	0
559390d15f001...	SELECT	a50a056c2b9e2...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d15ef36...	SELECT	a50a056c2b9e2...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d15f348...	SELECT	a50a056c2b9e2...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d15f6a4...	SELECT	91cc541703b93...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d15ebe0...	SELECT	a50a056c2b9e2...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d15f75e...	SELECT	12f232034e7d3...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d15fa17...	SELECT	12f232034e7d3...	95fb9e6dc37c2...	JDBC Thin Client	c10d233148484...	0	0
559390d65fd9a...	ETC	b8439ebae9ab1...	0	oracle@wsqlMa...	dee58bec1f68a...	0	0
559390d6600e2...	SELECT	fc00a3629c51cf...	0ee78a0564b10...	JDBC Thin Client	c10d233148484...	0	0
559390d65fa23...	SELECT	1038994c09761...	0	oracle@wsqlMa...	aeced65747928...	0	0
559390f263c92...	SELECT	fc00a3629c51cf...	0ee78a0564b10...	JDBC Thin Client	c10d233148484...	0	0
559390f263a23...	SELECT	fc00a3629c51cf...	0ee78a0564b10...	JDBC Thin Client	c10d233148484...	0	0
559391076632c...	SELECT	130a5ea45bcd4...	0ee78a0564b10...	JDBC Thin Client	c10d233148484...	0	0
559391256a35c...	SELECT	b4a6011772f53...	0	oracle@wsqlMa...	febdca0bc1eb9...	0	0
559391256a0bf...	SELECT	0dd0bebaa40c2...	0	oracle@wsqlMa...	febdca0bc1eb9...	0	0
559391256a842...	SELECT	a21a63c16e2b8...	0	oracle@wsqlMa...	febdca0bc1eb9...	0	0
559391256a1ed...	SELECT	a9bc0c04323d3...	0	oracle@wsqlMa...	febdca0bc1eb9...	0	0
5593912e6ae5a...	SELECT	2228cb0ee13f5...	0	oracle@DevAge...	dadd0f7cfc0801...	0	0
5593912e6ae98...	SELECT	2228cb0ee13f5...	0	oracle@DevAge...	dadd0f7cfc0801...	0	0

[그림 17] JSON 데이터를 데이터 테이블로 표현

8. 완성된 작품을 향하여 - Planche ERD

플란체는 구현된 바탕이 웹이다 보니 여러 다양한 자바스크립트 라이브러리를 쓰기가 수월하다. 현재는 플란체의 웹 기반 ERD 툴을 개발하고 있는데 해당 ERD 툴은 jsPlumb 을 사용하여 개발 중이다. 해당 기능을 추가하여 진정한 완제품 형태로 제작하는 것을 추후 목표로 하고 있다.



[그림 18] Entity 간의 관계를 쉽게 표현해주는 JSPlumb library

웹에서 데이터베이스부터 ERD 까지 모두 디자인 할 수 있는 툴 "플란체"가 가야 할 길은 아직 더 남았지만 앞으로 이 프로젝트를 발전시키기 위한 다양한 시도를 지속할 것이다.