Infrastructure and Web Applications for Application Programming Interface (API) of Bio-database, Ocean Data Bank (ODB) 海洋資料庫生物資料庫應用程式介面之基礎建構與網際網路應用

Chiyu Weng 翁其羽

2022-02-22

Table of Contents

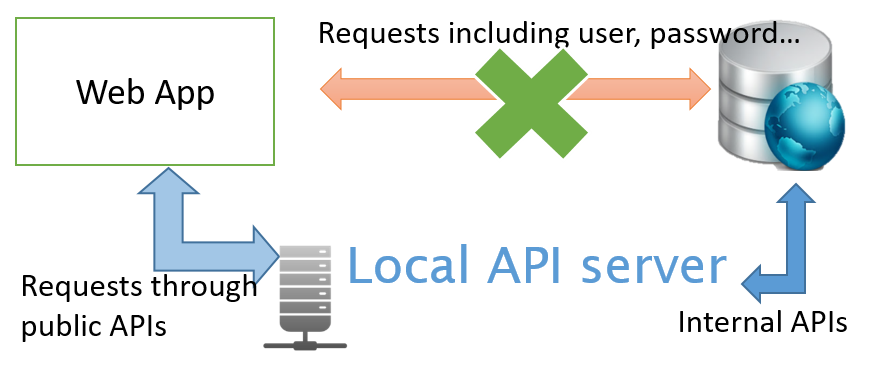
# Introduction

How can we design an infrastructure that help people easier, more safer, more convenient for adding ideas to use the bio-database of Ocean Data Bank (ODB)? ODB is dedicated to help academia to curate and use the databases whose data is mainly collected through marine research vessels which are supported by Ministry of Science and Technology (MOST), Taiwan. The data of ODB is in privacy under the restriction of the data release policies of MOST, and ODB compiles raw data after reviewing someone’s application. This restriction limits open usage of the information services we deliver. To bridge ODB’s information services between open usage and databases is a prerequisite, whereas a well-defined application programming interface (API) is often the answer ([Box 1.1](#box1-1)). In addition, ODB’s information services are usually not for implementing new theory or algorithm on scientific researches, but for helping academia to use data, check data patterns, and create data modeling in their researches. These considerations form the basis for how to construct the infrastructure of information services to use the bio-database of ODB:

Box1.1

✓ API is a set of interfacing specifications for machine-to-machine communications. Here we focus on web API, i.e., data transport upon HTTP request/response structure. Most popular web API protocol is REST (Representational State Transfer), and increasingly used, GraphQL API. Commonly used data formats are JSON and XML.

✓ API can provide a secure way to access the bio-database (internal APIs, Fig. ) with authority management. On the other hand, API can also provide open-access methods for public information compiled from the bio-database (public APIs).



**Figure** : Simplified schematic for web API of the bio-database

# **Re-structuring bio-database**

## *Mutating a new bio-database on PostgreSQL*

The bio-database of ODB had been constructed since 2009 and completely set-up on Microsoft SQL Server since 2015. The data curation is basically done by a Microsoft C# program. These protocols of data curation was documented in 圖輯. For developing an open cross-platform framework for web APIs of bio-database, first, we mutated a bio-database on PostgreSQL, i.e., exported from SQL Server, and then re-imported to a new database by using PostgreSQL.

Box2.1

Why not just use original Microsoft SQL Server as backend database for developing APIs of bio-database?

✓ SQL Server is a commercial software with annual license fees.

✓ SQL Server is suited for the Microsoft Server based framework, but not for an open cross-platform framework.

Why choose PostgreSQL?

✓ PostgreSQL is a powerful open-source object-relational database systems.

✓ PostgreSQL can be well integrated in Geographic Information System (GIS) applications by its PostGIS extension. - It’s an important reason why we choose PostgreSQL. Most of APIs in ODB are used to develop GIS related applications.

As a backend database for API in production, we do not need some columns of variables originally in SQL Server (Fig. 2a) that used to remark some quality control (QC) issues. So when mutating the bio-database on PostgreSQL, we also did some simplifications for database schema (Fig. 2b). Roughly speaking, we left the “remark\_xxx” or “flag\_xxx” columns in SQL Server only, and in PostgreSQL, re-arranged the tables to make frequently-used variables more easily be accessed in table “cast\_site” (site information) and “taxa\_data” (abundance data of taxonomic groups). Another important modification is to construct classification of taxonomic groups in table ’taxon\_group” that can help us in analytical statistics among taxonomic groups when using bio-database data. The major parts of schema for bio-database on PostgreSQL include: taxon\_group:

1. cast\_site:
2. c00\_cast\_tbl:
3. t00\_taxa\_tbl:
4. taxa\_data:

# NOW in Ubuntu, this driver is not support old SQL Serve 2008 R2. It only works in Windows  
library(RODBC)  
ms\_conn <- odbcDriverConnect(paste0('driver={ODBC Driver 18 for SQL Server};server=', sqlServerHost, ';database=', sqlServerDB, ';uid=', sqlServerUser, ';pwd=', sqlServerPass))

## Warning in odbcDriverConnect(paste0("driver={ODBC Driver 18 for SQL  
## Server};server=", : [RODBC] ERROR: state 08001, code -1, message [Microsoft]  
## [ODBC Driver 18 for SQL Server]SSL Provider: [error:1425F102:SSL  
## routines:ssl\_choose\_client\_version:unsupported protocol][error:140B40C7:SSL  
## routines:SSL\_do\_handshake:peer did not return a certificate]

## Warning in odbcDriverConnect(paste0("driver={ODBC Driver 18 for SQL  
## Server};server=", : [RODBC] ERROR: state 08001, code -1, message [Microsoft]  
## [ODBC Driver 18 for SQL Server]Client unable to establish connection

## Warning in odbcDriverConnect(paste0("driver={ODBC Driver 18 for SQL  
## Server};server=", : ODBC connection failed