# Processing data from an external database

For applications like the ParkingPermit application, some of the application’s data is in an external database CX. Other data is maintained in its own database to support the ‘business logic’ of the application.

The ParkingPermit application needs the following data from CX:

* The StudentId or FacultyId. (The UF Id)
* The first and last name of the student or faculty/staff member registering for a parking permit

This data *infrequently* changes.

There are a couple concerns about querying data from an external data store, such as CX.

One concern is the performance. The second concern is application location.

Performing a query across the ODBC link each time the application needs to look up the UFID and name imposes a performance overhead. It also increases the risk of failures and dependencies of the CX database being online and available.

By requiring the Web Application to use an ODBC connection to an external database, independence of the application location is inhibited. Web Applications should be location agnostic. That is, it should be as easy to deploy a Web Application to the cloud or to a third party hosting service as it is to deploy to on on-premises server.

A Web Application with a dependency of an ODBC connection configured on a server to connect to an Informix database breaks location independence.

# Methods to Achieve Location Independence

So a couple ways often seen implemented to achieve location independence.

One is using a proxy service. The other is using a remote service to push changes to the Web Application.

## Proxy Service

A proxy service is a service resides on an on-premises server and used by the Web Application to obtain data from the external data source.

When data is frequently changing, a proxy service is a good solution because it does the queries to get the latest data on behalf of the Web Application.

The Web API service would be hosted on a machine with the ODBC configured connection and make the low level API calls to the CX database. The proxy service hides the ugliness of the CX table queries from the consumer.

The downside of a proxy service is the web app needs to make two hops to get the data. One hop to the proxy service. The proxy service needs to make a second hop to query the external (CX) database.

But, is a Web API service a good solution for *infrequently* changing data?

Possibly, but there are a several downsides.

To avoid the network hop to the CX database, the proxy service could periodically read the CX data, cache the data, and serve it to the Web Application. So a caching mechanism needs developed.

Another downside is the Web Application still needs a network hop to the proxy service to obtain the data. Although the performance lag when the Web Application resides on premises in minimal, the delay become more significant when the Web Application is hosted in the cloud or third party hosting service.

Another downside is the need to creating DNS entries and opening ports to the on premises proxy service.

## Remote Web API service to Push Changes

The proxy service is close in location to the external database. A remote Web API service is close in location to the Web Application. Using the tooling in Visual Studio, the remote Web API service becomes part of the application and is deployed as part of the Web Application.

The purpose of the Web Api is to provide updates and deletes to table(s) in the Web Application’s database, and keep the Web Application’s table(s) in sync to the *infrequently* changing CX data.