A. Abstract: A client that has 3 shops asked me – ‘can I have one  
program, across all three locations? So, that people who join the program in  
shop A will be able to continue accumulating points when they shop in B and  
C?’

B. Current limitation:

Currently our database design and BL doesn’t allow for it as we essentially  
have 1-1 relationship between Company and Program. The fundamental flow with  
this assumption is that the Company can have multiple shops – let’s call  
them Location (to be consistent with our terminology).

prg\_program.CompanyID is FK for com\_company.ID

C. Proposed solution:

Introducing com\_location table. Location table will essentially be a mapping  
table that can accommodate for the following cases:

1. One company has 1 shop and 1 program;

2. One company has 2 shops and wants each to be on a separate program;

3. One company has 2+ shops and wanted to use same program;

4. One company has 10 shops and wanted 5 of them to be on same program,  
while reminder of 5 on their individual programs;

5. One company has 11 shops and wanted 5 of them to be on program (A),  
next 3 on program©, next 2 on program (D) and last one shop on program  
(E);

6. One company has 3 programs that run concurrently;

7. One company has 5 shops. 2 shops share 2 programs between them (each  
has access to two programs), 3 shops sharing one program;

com\_location table structure:

ID int (PK)

TypeID int FK to com\_location\_type

CompanyID int FK to com\_company

ProgramID in FK to prg\_program

com\_location\_type table structure:

ID int (PK)

Description varchar (16)

– This is a control table with following Descriptions:

o Single (1 location has 1 program OR 1-to-1, to accommodate for case C.1  
and C.2 above);

o Multi (Multiple locations have 1 program OR many- to-1, to accommodate  
for case C.3 above);

o Multi-Reversed (1 location has multiple programs OR 1- to-many, to  
accommodate for case C.6 above);

o Multi-Extended (More than one location has More than one program OR  
many- to-many, to accommodate for case C.4, C.5, C.7 above).

D. Concerns:

1. Database access speed and size optimization. We have to make sure  
that the implementation doesn’t slow down our current program fetching  
algorithm;

2. Number of modifications to accommodate for new DB structure and  
business logic in

a. WebApp;

b. MobApp (iOS) – shouldn’t be affected as WebApp provides it with XML  
with all nodes populated;

c. MobApp (Android) – shouldn’t be affected as WebApp provides it with  
XML with all nodes populated;

d. Backwards compatibility (if we release this in 1.0.8 then it  
shouldn’t break 1.0.7 or 1.0.6 apps).

Before we start implementing this – I’d like your input with thoughts if  
this could be done better or more efficiently from D. point of view. Also  
this is a major change to the database structure, albeit very necessary for  
wider Company’s adoption (think about McDonalds franchise that wants to run  
same program for all 150 stores in the country).

Update on Location DB design

1. At the moment the limitation is that Longitude and Latitude are in Company table. This doesn’t cater to the scenario when a company has more than one Location (i.e. one company 3 shops)
2. Introduction of Location table as on diagram below) will add ultimate flexibility to LAZ – so we can handle any combination in the future.
3. Before when we used in MobApp ‘Find Program’ function – WebApp would fetch Company-Program. With new design it will fetch Company-Location-Program (taking the actual coordinates from Location table)
4. Before when we used in MobApp ‘Find Coupon’ function – WebApp would fetch Company-Coupon. With new design it will fetch Company-Location-Coupon (taking the actual coordinates from Location table).

Migration process

1. Timing 9pm NZT to 6am NZT on Monday
2. Backup both DB and PHP
3. Enable ‘Maintenance Mode’ (Website will display ‘maintenance mode’ message; MobApp – will show a popup ‘Server is in maintenance mode. Please check again in an hour’ message when Customer tries to Find Program/Coupons.
4. Apply DB structure update – enable index insertion, UPDATE, INSERT, etc
5. The SQL script needs to be created and tested on /test – that does complete DB update and insert new values, populating Location table with appropriate values from Company table (Long/lat) for location. For example French Bakery – will have one Company located in Browns Bay; three locations with their Names [Browns Bay, Milford, Takapuna); with their respective tags.
6. Apply and enable Indexes and Foreign Keys, check DB integrity.
7. Deploy PHP script. Test that all works by creating new company with multiple locations

Usage scenarios

As you suggested – this only will be available for Local Plus and higher plans.

Add new left side menu item – Locations – it will list all Company locations and if clicked on individual Location name – will enable Location edit (Name, Tag, Description)

Program List – taping on individual Program – will show them program details. AT right top will be ‘Show Locations’ menu item – if clicked – will show list of Locations

At top left will be menu item ‘Associate Location’ – clicking on it will allow to associate already created Location to current Program.

Company Details screen will have at top ‘Add Location’. (Defaul location will always be created upon creation of the Company – utilizing Long/Lat using Company Address) Clicking on this will allow to add second and sub sequential locations to the company. Number of locations will be limited by current Plan.

Location

ID smalint PK

Name (Move from **Program)**

Tag (Move from **Program)**

CompanyID FK (Company)

ProgramID FK (Program)

CouponID FK (Coupon)

Long (Move from **Company)**

Lat (Move from Company)

Coupon

Program

Company Group

Company