COMP. 6108

Assignment 1 - Part 2

**George Czabania**

*27002246*

*23rd October 2013*

Table of Contents

[Introduction 3](#_Toc370304786)

[Logical Design 4](#_Toc370304787)

[The Integrated Normalised Entity-Relationship Diagram 4](#_Toc370304788)

[Physical Design 5](#_Toc370304789)

[Relations 5](#_Toc370304790)

[Data Volume Map 6](#_Toc370304791)

[Data Usage Maps 7](#_Toc370304792)

[All Students and Matching Vacancies 7](#_Toc370304793)

[All Students Interested in One Vacancy 8](#_Toc370304794)

[Considerations for De-normalisation 9](#_Toc370304795)

[Joining Tables to create a One-To-One relationship 9](#_Toc370304796)

[Horizontal partitioning 10](#_Toc370304797)

[Vertical Partitioning 11](#_Toc370304798)

[Indexing 11](#_Toc370304799)

[Final Entity Relationship Diagram 12](#_Toc370304800)

[Data Dictionary 13](#_Toc370304801)

[Location 13](#_Toc370304802)

[Employer 13](#_Toc370304803)

[Vacancy 13](#_Toc370304804)

[Student 14](#_Toc370304805)

[StudentEmployed 14](#_Toc370304806)

[Skill 15](#_Toc370304807)

[StudentSkill 15](#_Toc370304808)

[VacancySkill 16](#_Toc370304809)

[Applicant 16](#_Toc370304810)

[Appendix 17](#_Toc370304811)

[Communications with client 17](#_Toc370304812)

[Part 1 17](#_Toc370304813)

[Part 2 17](#_Toc370304814)

# Introduction

In this report, I will take the logical design of the database from Part 1 of the Assignment and convert it into a physical design and implementation.

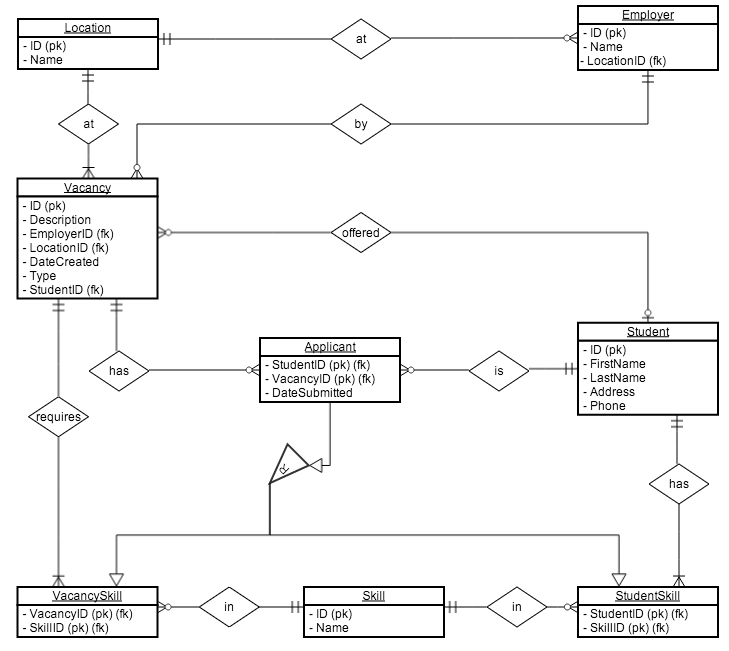
This process will include modeling the data volume and access frequency for the two most important processes as well as considering possible de-normalisation methods to improve performance.

This will result in an SQL script file that can be used to generate the tables and populate them with example data.

# Logical Design

## The Integrated Normalised Entity-Relationship Diagram

This is the integrated normalized entity-relationship diagram from Part 1 of the Assignment.



# Physical Design

## Relations

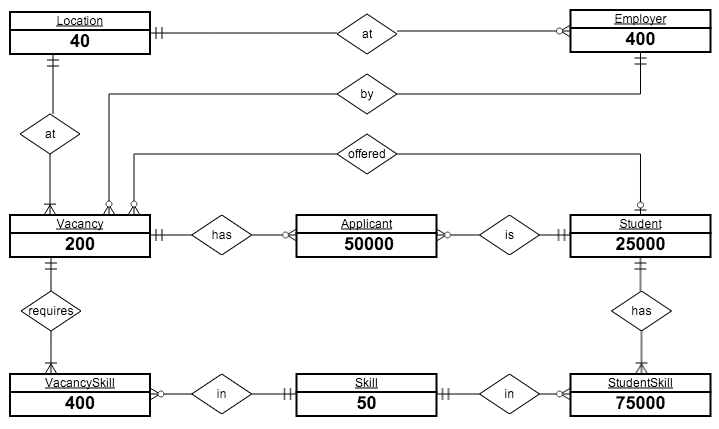
**Tables**

* Location ( ID, Name )
* Vacancy ( ID, Description, EmployerID, LocationID, DateCreated, Type, StudentID )
* Employer ( ID, Name, LocationID )
* VacancySkill ( VacancyID, SkillID )
* Skill ( ID, Name )
* StudentSkill ( StudentID, SkillID )
* Student ( ID, FirstName, LastName, Address, Phone )
* Applicant ( StudentID, VacancyID, DateSubmitted )

**Foreign Keys**

* Vacancy
  + EmployerID: ON DELETE NO ACTION
  + LocationID: ON DELETE NO ACTION
  + StudentID: ON DELETE SET NULL
* Employer
  + LocationID: ON DELETE NO ACTION
* VacancySkill
  + SkillID: ON DELETE CASCADE
  + VacancyID: ON DELETE CASCADE
* StudentSkill
  + SkillID: ON DELETE CASCADE
  + StudentID: ON DELETE CASCADE
* Applicant
  + StudentID: ON DELETE CASCADE
  + VacancyID: ON DELETE CASCADE

## Data Volume Map



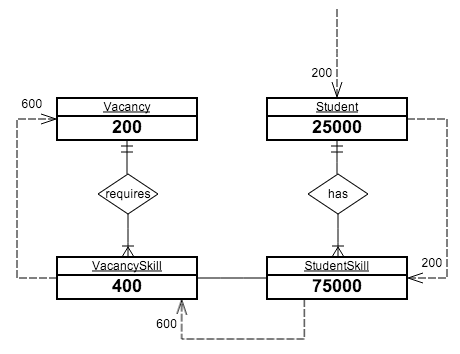
These estimated volumes were gathered from the client:

* Location: 40
* Employer: 400
* Vacancy: 200
* Student: 25000
* Skill: 50
* Applicant: (2 per Student) = 50000
* VacancySkill: (2 per Vacancy) = 400
* StudentSkill: (3 per Student) = 75000

## Data Usage Maps

### All Students and Matching Vacancies

This process is run 200 times a day.

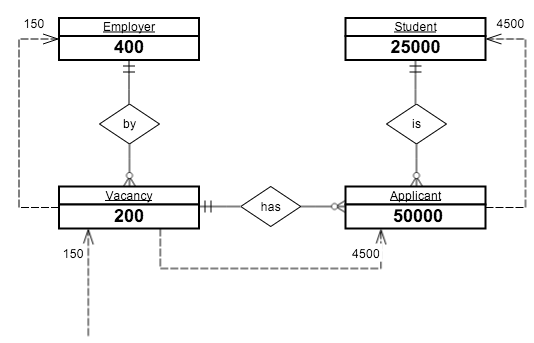


Assumptions:

* For each request for a Student, one request will need to be made to get a list of their Skills.
* On average, Students have three Skills. For each of these Skills, we will need to check it against every VacancySkill and match the SkillID attribute.
* For each VacancySkill we need to get the Vacancy, this takes a single request per VacancySkill.

### All Students Interested in One Vacancy

This process is run 150 times a day.



Assumptions:

* For each Vacancy we need to send one request to get the Employer.
* On average, a Vacancy will have 30 Applicants. For each of these Applicants, we need to send a request (30 \* 150 = 4500).
* For each Applicant we need to send one request to get the Student details of who applied.

## Considerations for De-normalisation

These considerations will be based on the two main processes of the system:

* All Students and Matching Vacancies
* All Students Interested In One Particular Vacancy

### Joining Tables to create a One-To-One relationship

This de-normaliztion method is used to increase performance by reducing the number of joins that must be done when querying information across multiple tables.

This could be useful for the All Students and Matching Vacancies process to join the Vacancy and VacancySkill tables, as vacancies only have two required skills on average but are often needed to be queried together.

To turn this into a one-to-one relationship, we could merge VacancySkill and Vacancy together. This would duplicate a lot of data, but would possibly improve performance.

**Vacancy**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **s** | **Description** | **EmployerID** | **LocationID** | **DateCreated** | **Type** |
| 1 | Cashier | 2 | 3 | 2013-10-22 | 0 |

**Skill**

|  |  |
| --- | --- |
| **VacancyID** | **SkillID** |
| 1 | 10 |
| 1 | 20 |
| 1 | 30 |

**Vacancy + Skill**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **EmployerID** | **LocationID** | **DateCreated** | **Type** | **SkillID** |
| 1 | Cashier | 2 | 3 | 2013-10-22 | 0 | 10 |
| 1 | Cashier | 2 | 3 | 2013-10-22 | 0 | 20 |
| 1 | Cashier | 2 | 3 | 2013-10-22 | 0 | 30 |

As you can see each vacancy is duplicated for every skill it requires. This makes maintaining it a lot harder, as rows could easily get out of sync if only one instance of them is changed.

**Advantages:**

* Less joining needs to be done in the SQL query (which is a relatively slow process).

**Negative:**

* Duplication of data takes up more space
* Added complexity – changing a vacancy now takes longer because it needs to update all instances in the database.
* There is also the possibility of data-corruption if one instance of the vacancy is set to something different and doesn’t update the other instances. So one vacancy might get a student to apply for it, but not applied to the other instances of the same vacancy

### Horizontal partitioning

Horizontal partitioning is used to distribute the tables rows.

A possible use for is that there are expected to be 25000 of students stored in the database, but we can assume that only those who aren’t employed need to regularly accessed. To improve performance, the Student table could be horizontally partitioned depending on whether the student is employed or not. This would create two tables:

* Student
* StudentsEmployed

This would greatly increase performance, as the system has fewer records to sort through when looking for a student. We could also do it in a way that would minimize complexity by not storing skills for the employed students, as they are no longer needed.

### Vertical Partitioning

This involves splitting up tables by distributing the columns of a table into several separate tables. Each table needs to store the primary key to be able to match the records together.

A possible use case is that Student table is accessed very often to get each students first and last name. The student table also stores the phone number and address of each student, but this information is accessed only on rare occasions. To improve performance, the student table could be vertically partitioned to create two tables:

**Student**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **StudentID** | **FirstName** | **LastName** | **Phone** | **Address** |
| 2 | George | Smith | 123 | 20 Round St |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Student\_Name** | | |  | **Student\_Contact** | | |
| **StudentID** | **FirstName** | **LastName** |  | **StudentID** | **Phone** | **Address** |
| 2 | George | Smith |  | 2 | 123 | 20 Round St |

This improves performance for queries that only get the name of the student, or the phone and address. But slows down performance for queries that need both the name and phone or address. This also makes managing foreign keys harder, as you have to check referential integrity on both tables.

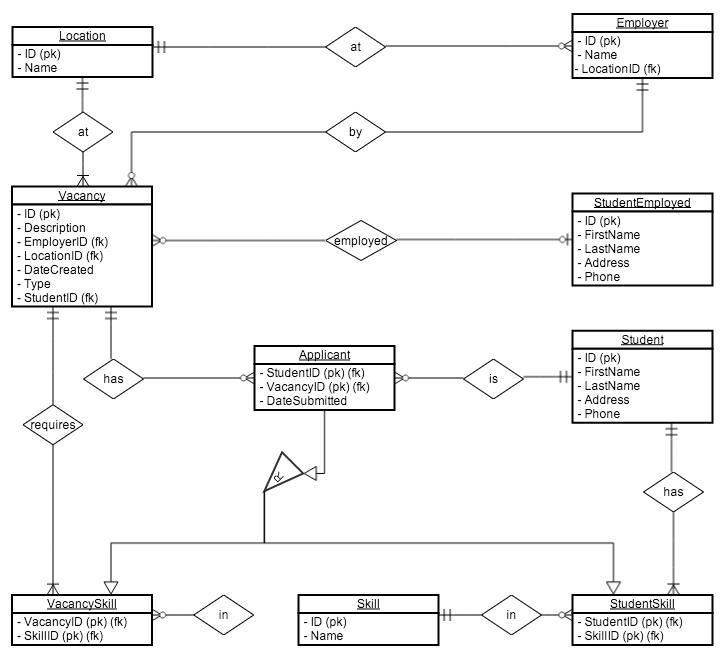
This is not an ideal method to use on this system, as phone and address data are not used in the two main processes. And moving them into a separate table doesn’t help us.

### Indexing

This involves creating indexes for columns that are not already indexed by default (such as primary keys). Most of the time this means indexing foreign keys, as these are often used to locate records in a table. However for both of the important processes, every attribute that is accessed is already a primary key and has been indexed by default by the DBMS. The one exception is Vacancy.EmployerID.

We could index this field, but it wouldn’t help in either of the two processes, as we are locating each vacancy by its primary key.

## Final Entity Relationship Diagram



The only change made was to horizontally partition the Student table and create the StudentEmployed table. This new table now holds all the students are employed at a Vacancy.

## Data Dictionary

### Location

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| ID | Location ID | int |  | 12 | ✔ | ✔ | ✔ |  |  |
| Name | Location Name | varchar | 30 | Lynmore | ✔ |  |  |  |  |

### Employer

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| ID | Employer ID | int |  | 48 | ✔ | ✔ | ✔ |  |  |
| Name | Employer Name | varchar | 30 | Wairiki | ✔ |  |  |  |  |
| LocationID | Employer location | int |  | 2 | ✔ |  |  | ✔ |  |

### Vacancy

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| ID | Vacancy ID | int |  | 15 | ✔ | ✔ | ✔ |  |  |
| Description | Vacancy Description | varchar | 30 | Lynmore | ✔ |  |  |  |  |
| EmployerID | Vacancy Employer | int |  | 32 | ✔ |  |  | ✔ |  |
| LocationID | Vacancy Location | int |  |  | ✔ |  |  | ✔ |  |
| DateCreated | Date Vacancy was created | timestamp |  | 2013-10-22 18:45:00 | ✔ |  |  |  | CURRENT TIMESTAMP |
| Type | Full time or part time | tinyint |  | 0 | ✔ |  |  |  |  |
| StudentID | The Student employed | int |  | 30 |  |  |  | ✔ |  |

### Student

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| ID | Student ID | int |  | 2732 | ✔ | ✔ | ✔ |  |  |
| FirstName | Student First Name | varchar | 30 | Jim | ✔ |  |  |  |  |
| LastName | Student Last Name | varchar | 30 | Smith | ✔ |  |  |  |  |
| Address | Student Address | varchar | 100 | 19 Devon St Rotorua | ✔ |  |  |  |  |
| Phone | Student Phone Number | varchar | 20 | 021 793 082 | ✔ |  |  |  |  |

### StudentEmployed

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| ID | Student ID | int |  | 2732 | ✔ | ✔ | ✔ |  |  |
| FirstName | Student First Name | varchar | 30 | Jim | ✔ |  |  |  |  |
| LastName | Student Last Name | varchar | 30 | Smith | ✔ |  |  |  |  |
| Address | Student Address | varchar | 100 | 19 Devon St Rotorua | ✔ |  |  |  |  |
| Phone | Student Phone Number | varchar | 20 | 021 793 082 | ✔ |  |  |  |  |

### Skill

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| ID | Skill ID | int |  | 2 | ✔ | ✔ | ✔ |  |  |
| Name | Skill Description | varchar | 20 | Word Processing | ✔ |  |  |  |  |

### StudentSkill

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| SkillID | Student Skill | int |  | 12 | ✔ | ✔ | ✔ | ✔ |  |
| StudentID | Student ID | int |  | 23 | ✔ | ✔ | ✔ | ✔ |  |

### VacancySkill

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| SkillID | Vacancy Skill | int |  | 52 | ✔ | ✔ | ✔ | ✔ |  |
| StudentID | Vacancy ID | int |  | 123 | ✔ | ✔ | ✔ | ✔ |  |

### Applicant

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Data Type** | **Size** | **Example** | **Required** | **Indexed** | **Primary Key** | **Foreign Key** | **Default Value** |
| StudentID | Student ID | int |  | 12 | ✔ | ✔ | ✔ | ✔ |  |
| VacancyID | Vacancy ID | int |  | 32 | ✔ | ✔ | ✔ | ✔ |  |
| DateCreated | Date Application Created | datetime |  | 2013-10-22 19:42:20 | ✔ |  |  |  | CURRENT TIMESTAMP |

# Appendix

## Communications with client

### Part 1

1. How many entries do you expect to store in each of these tables?

* Locations: **40**
* Employers: **400**
* Students: **25000**
* Skills: **50**
* Applicants (for students who have applied for a vacancy): **Applicants are students (on average 2 applications per student)**
* Student Skills (for skills that students have): **On average a student has 3 skills. You may calculate this.**
* Vacancy Skills (for skills required by vacancy): **On average a vacancy requires 2 skills. You may calculate this.**

1. How often will each of these tables be accessed (per hour)?

* **Limit this to the specified processes (two). The frequency of usage of such processes is mentioned in the assignment.**

### Part 2

1. How many entries do you expect to store in each of these tables?

* Vacancies: **200**

1. On average, how many applicants are there for each vacancy?

* **30 applicants for one vacancy.**