Research Higher Degree Database

Operational Database System Documentation

Prepared by:
Thursdays <u>Group 4</u>

Sam Deane dean0109

Andrew Zschorn zsch0003

Version 1.0

12/06/2014

Created as part of the requirements for Advanced Database GE 2014

Contents

1 Requirements

Introduction

Document Purpose

The purpose of this document is to introduce and specify the requirements of a new software database (“the database product”).

The database product will constitute only part of a larger human-computer software system responsible for developing Research Higher Degree (RHD) applications. The database is to assist staff of the School of Computer Science, Engineering and Mathematics (CSEM) of Flinders University in pursuing this task.

Document Conventions

Throughout this document the terms ‘product’ and ‘database’ are used interchangeably as the product consists solely of a single database.

Requirements can be inherited and composed or be aggregates of other requirements.

Abbreviations will be used with definitions available in an appended glossary at the end of the document.

Intended Audience and Reading Suggestions

This document is intended to be read by Developers, RHD office staff members and CSEM faculty staff. Developers and RHD Staff Members are suggested to read this document in its entirety in the form presented herein. CSEM staff should read the mission statement, product features, product perspective and the specific system requirements to gain a simpler overview of the proposed product.

1.1 Fact Finding

1.1.1 Fact-finding techniques

The requirements will be gathered through documented information received from the Research Higher Degree Admissions officer as well attained through follow up meetings and via FLO discussion boards to clarify any issues or attain more detailed information on some areas.

This is anticipated that there will be an opportunity for the users/customers to refine their understanding of the requirements as more implementation ideas and details are presented to them by our group and others. The users/customers in this case are Associate Professor Paul Calder as the Director of Higher Research Degree Studies in CSEM and Dr Denise de Vries as a member of staff who responds to many unsolicited enquiries about HRD.

The Flinders University Office of Graduate Research accepts form-based applications (Flinders University Office of Graduate Research 2014b) that all successful HRD applicants have to go through. The requirements of this existing system can be used to help define the checklist of documents and other outputs of our new system.

This system will have to store information about areas of research. There is an existing scheme in use in academia in Australia called the Field of Research codes (Australian Bureau of Statistics 2014). Flinders University staff web pages contain examples of how academics describe their research interests. The Flinder's University "Find a Supervisor" website (Flinders University Office of Graduate Research 2014a) demonstrates how areas of research can be mapped to potential academic supervisors.

The CSEM school website shows the list of RHD awards they run.

An applicant's Grade Point Average (GPA) is crucial for making decisions about the viability of their application. A complexity here is that GPA is calculated in different ways between national tertiary education systems and even between different compatriot institutions. We need to support automatic standardisation of applicant's GPA scores in our system to simplify the process of taken decisions about the viability of applications. Thus we need to build in a level of awareness of the scoring systems. Wikipedia has a detailed page about GPA scoring in Australia (Wikipedia 2014a) and an overview of systems in many other nations (Wikipedia 2014b). All particular information contained on these pages will need to be confirmed directly with different institutions, but these pages give a good overview of the diversity of scoring systems, which we can use to inform the scope of this aspect of our database.

1.1.2 Purpose

1.1.2.1 Product Perspective

This database system is a novel product. Email is currently the only tool that is used to help perform RHD management; however it does not provide any help specific to this purpose. The new product will be specifically designed to the task of managing RHD applications and will provide assistance where possible to aid in this task.

However, the new product won't replace or constrain email communications. Rather, the system will support CSEM staff in managing RHD applications, and make the mail communications quicker to write and easier to keep track of.

There is a form-based application process run by Flinders University Office of Graduate Research. This constitutes a second phase of the application process, which each successful applicant must go through. A goal of our system is then to help CSEM staff make applicants aware of and be well prepared for the requirements of this subsequent, form-based, formal application process. We use the term 'elevate' to describe transitioning a promising application from being managed within our database to using the form-based, formal application process.

1.1.2.2 Product Features

To create a Database university staff use to track RHD applications at an early stage, i.e. before the applications are managed by the more formalised form-based system run by the Office of Graduate Research. The project is limited to the creation of a usable full functional database that can be accessed through the command line. No Graphical User interface GUI, will be required for this stage of development. However initial requirements are hinting towards an integrated add-on to existing email clients (e.g. Microsoft Outlook).

1.1.3 Mission statement

To support CSEM staff working together to quickly select and guide high-quality Research Higher Degree applications from initial, direct, informal contact to the formal university RHD application process.

1.1.4 Scope and boundaries

Figure 1 on page 7 shows the current and future human/computer system for managing RHD applications in the school.

The project is limited to the creation of a usable full functional database that can be accessed through the command line. No Graphical User interface GUI, will be required for this stage of development. The main benefit of the product is to provide a place to store the core details of an application in an easily viewable and editable form that will enable application and applicant data tracking over the evaluation of one or more applications. This database will give staff a greater understanding of prospective students allowing the strengths of the university and future student to be combined whilst continuing the expansion of the CSEM faculty as outlined by the CSEM mission statement. In the short term the system will improve the quality of application screening and positioning.

1.1.4.1 Operating Environment

The database itself will run on a server within the CSEMs IT department. Since the Server is not expected to be used heavily (on the order of a 200 or so transactions and simple queries per day) it is not expected that any new hardware will be required. The product will be built as a MySQL database which is freely available to all enterprises so there will be no additional cost for the database software itself.

1.1.4.2 Design and Implementation Constraints
The product will conform to standard MySQL core packages to enable greater update and management flexibility. Some security precautions will be taken to ensure that the data is not available to students and staff from other faculties.

1.1.4.3 User Documentation
Conceptual, logical and physical diagrams of the database will be delivered along with instructions on maintaining the database will be delivered with the product in the form of a PDF or E-Manual.

1.1.4.4 Assumptions and Dependencies
It is assumed that all users and the database itself have read/write access to a common file system area, for the purpose of linking documents such as PDFs or image scans to applications.

An associated front end application for users to interface with database has not yet been designed. It is expected that once it has been analysed and requirements gathered then small modifications to the database are expected to be made, predominately in the form of new queries. External Interface Requirements

1.1.4.5 User Interfaces
For the scope of this software database product, specific requirements for the user side of any user interface won't be considered.

However, the database will contain tables, views, stored queries and data examples that would be appropriate to drive the information content for an appropriate UI, given the feature requirements specified above. This will be demonstrable through direct database operations, expressed in SQL statements and responses.

1.1.4.6 Software Interfaces
This product will be delivered as a standalone software system. However it is expected that a front end application will be developed to interface with the database in future.

1.1.4.7 Communications Interfaces
The connection interfaces are expected to be determined by the server on which the database resides in addition to the MySQL servers own settings. These will be determined once further information about the rollout of the database is realised.

1.1.4.8 Initial database size
There is only a need to record currently active applications, not to initially load up the database with the history of completed applications. Hence, in this case the rate of growth is far more important than the initial size.

1.1.4.9 Database rate of growth
There are 40 members of CSEM staff and most would be contacted by 20-40 potential applicants during peak periods. Of these, at least one third would be immediately judged unsuitable and never be recorded in our system.

1.1.4.10 Types and average number of record searches
Most frequent searches are expected to be recalling applications that have been worked on recently. This may be run 100 times per weekday. The next most frequent search would be searching for applications given matching Field of Research codes. This may be run 30 times per weekday. And the last significantly frequent search is searching for applicant record for record by applicant name.

1.1.4.11 Networking and shared access requirements
We need to support at least 10 staff using the application simultaneously.

1.1.4.12 Performance
All searches, single record inserts and single record updates should return within one second.

1.1.4.13 Security
Access to only the information a staff member requires to do their job will be allowed. Views will be used to restrict operations available to certain users and groups of users. We are also assuming that all users have undergone training in professional integrity, so, it is expected there will not be any misuse of the product

1.1.4.14 Backup and recovery
The database should be backed-up by standard CSEM IT department infrastructure every week night. Recovery procedures should be tested every three months to ensure they can be relied upon.

1.1.4.15 Legal issues
Of primary concern here is applicant privacy, due to the collation of personal information. Access will have to be logged and shown to always be in the course of work duties.

1.1.4.16 Software Quality Attributes
Currently there are no specific quality attributes the software hopes to achieve. These general quality attributes will be dictated by the underlying MySQL server

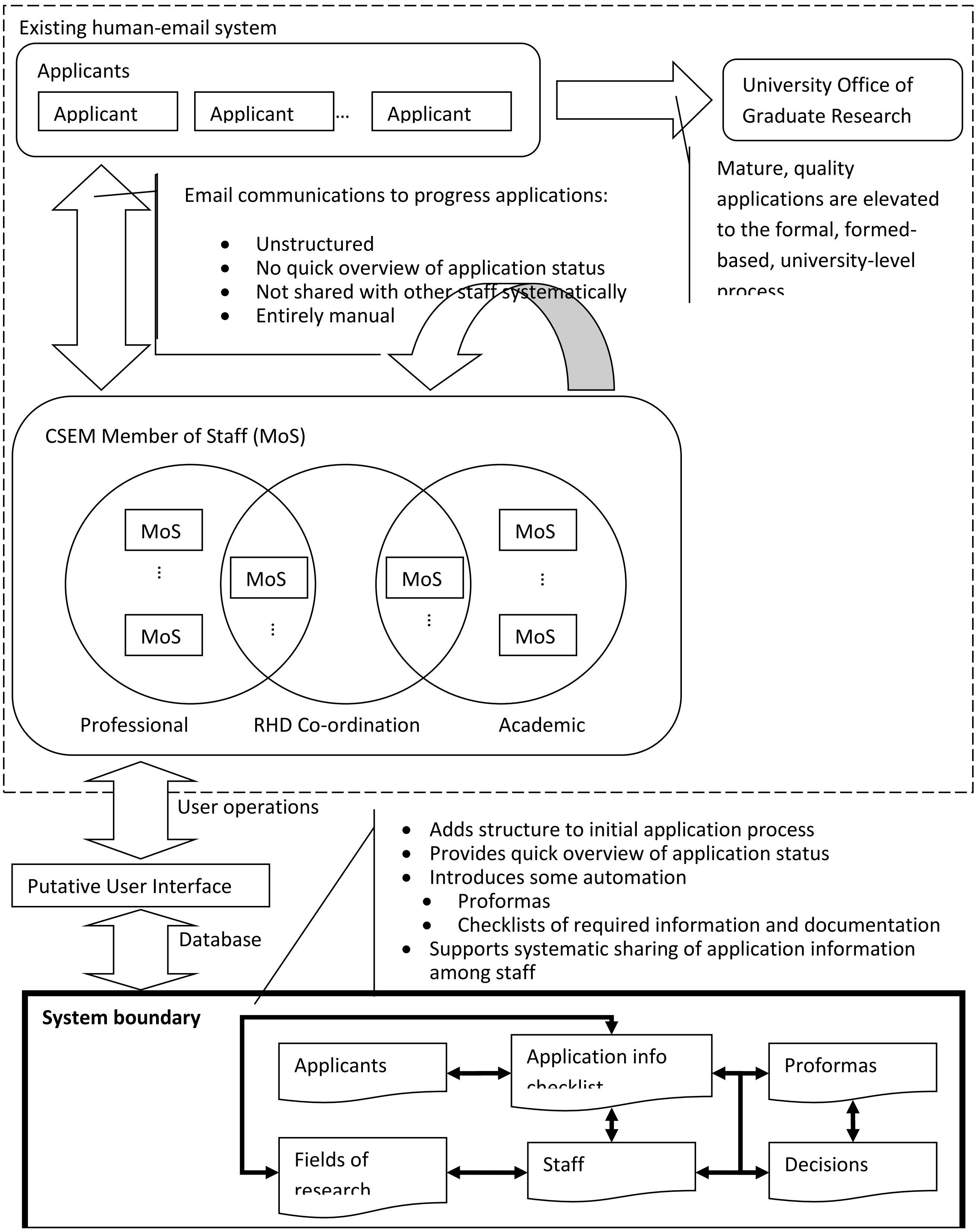

Existing human-email system
Applicants
Applicant
Applicant
...
Applicant
University Office of Graduate Research
Mature, quality applications are elevated to the formal, formed-based, university-level process
Email communications to progress applications:
Unstructured
No quick overview of application status
Not shared with other staff systematically
Entirely manual
CSEM Member of Staff (MoS)
MoS
MoS
MoS
MoS
MoS
MoS
Professional
RHD Co-ordination
Academic
User operations
Adds structure to initial application process
Provides quick overview of application status
Introduces some automation
Proformas
Checklists of required information and documentation
Supports systematic sharing of application information among staff
Putative User Interface
Database
System boundary
Applicants
Application info checklist
Proformas
Fields of research
Staff
Decisions

Figure 1 Project Scope Diagram. The new database software product shown in context of the wider human-computer RHD application system

1.1.5 Target users and audience

The user classes for this system are CSEM Professional Staff, CSEM Academic Staff, and CSEM RHD Co-ordination Staff. Note that members of the CSEM RHD Co-ordination Staff are all also ether Professional or Academic Staff.

A common characteristic shared by all staff members is that they don't have a lot of time available to help develop RHD applications. Thus it's critical that the database product provide the ability to quickly get a picture of an application's status, and prompts of how best to progress the application from that point.

Another possible user class could be those Academic Staff who have expressed an interest in supervising an applicant's project. However, as we expect there will be frequent, perhaps even ambiguous, shifts into and out of this user class, and we don't want to create a lot of work for users to have to update this information continuously, we chose not to distinguish this class in terms of their feature requirements.

Another possible user class could be the Applicants themselves. We can see clear benefits if the applicant were able to access their application directly, as they would be able to add any outstanding information themselves directly. This would avoid the requirement of emailing questions and information to staff members. However, for the purposes of this initial stage of development we won't consider the applicants as a user of this system. We leave that to a subsequent iteration of this system's development.

1.1.6 User views

The database has three user groups

These are

- Views for all staff, including professional, academic and RHD staff
    - Require a view to list all applications they are currently working on.
- Academic staff view
    - Views to help them search for new, relevant RHD applications
- Views for RHD Co-ordination staff:
    - Views to help keep track of all current applications, and which staff are involved in them

1.2 Requirements collection and analysis

Here we list the functional requirements for each group of users. It should be noted that the database will be designed in such a way as to make these user functions easily achieved through a user interface, which is not yet designed. The database won't be able to support these features directly in a way suitable for normal users; only expert users would be expected to be able to drive the database to achieve the functions.

1.3 Professional Staff Functional Requirements

1. Create a new applicant record (broken down into smaller requirements)
    - An applicant will include the following, which also constitutes a checklist of information that applications will need to satisfy over time
        - Version ID
        - Each version will contain:

        1.1 Basic details
        - Full name
            - First name
            - Middle names
            - Last name
        - Date of birth
        - Sex
        - Flinders University student ID if applicable
        - Address
            - Country
            - State
            - Suburb
            - Postcode
            - Street and number
        - contact details
            - email
            - home phone (including country and areas codes)
            - mobile phone (including country and areas codes)
        - GPA
            - standardized or at least fully described

        1.2 history
        - most current CV
            - includes the Date of upload
        - Publications
            - Name of publication
            - Issue number
            - Issue date
            - Online link (if available)
            - Upload (if rights are ok)
            - Associate authors
        - Degrees completed
            - Degree title
            - Undergrad/postgrad
            - If postgrad is RHD
            - Institution
                - Name
                - country
            - year commenced
            - year completed

        1.3 citizenship details
        - Australian or New Zealand citizen or permanent resident;
        - Visa details (if none of the above is the case)
            - Country of origin of visa application
            - Visa status
            - passport scan attached
        - year of entry into Australia if applicable

        1.4 English language proficiency
        - IELTS/TOEFL
        - main language spoken at home

        1.5 Referees > 2
        - Name,

            - position,
            - phone number
            - email address
            - academic link (linked-in / University webpage)

        1.6 Record meta-data
            - Name of staff member who created the record
            - Timestamp of record creation

2. Create a new application record
    - Includes create applicant if does not exist
    - An application will include the following
        - Version number
        - Areas of research, for each
            - A list of Field of Research codes
            - A list of user-defined keywords
        - award sought (MSc, PhD etc)
            - Doctor of Philosophy, Master of Engineering, Master of Science (Computer Science), Master of Science (Mathematics).
        - research proposal summary (~100 words)
        - research proposal extended (~500 words)
        - proposed date of commencement
        - part-time/full-time
        - internal/external
        - proposed supervisors
            - telephone (from staff page, unique-ish ID)
            - name
            - position
        - proposed funding method
            - scholarship
            - financial guarantee
3. Use a proforma to generate standard correspondence
    - E.g. acknowledgments of receipt, requests for more information from checklist, etc.
4. Add records of correspondence concerning the application
    - E.g. acknowledgment of receipt, requests for more information from checklist, etc
    - Date of correspondence
    - To
    - From
    - Optionally attach document or image files
    - comments
5. Attach documents to an application. This includes adding descriptions of the document contents, including creating links to entries in the checklist that the document provides.
6. Delete an application
7. Delete an applicant
8. Delete a correspondent entry
9. View an application
    - Retrieve documents associated with applications
    - Most current version (but with links to previous versions)
10. View an applicant
    - Includes the display of associated applications
11. View application versions
12. View application changes
13. Search applications
14. Search applicants
15. Edit application (add new version - if significant change?)
16. Edit applicant (as hard copy changes)
17. Automatic generation of outbound correspondences (acknowledgement/RFI) appropriate to a particular applicant
18. View a checklist of the recorded and outstanding information require of an application
    - Entries not completed in the applicant, application includes requirement

- All information in the checklist can be described with:
  - A status: unknown/unstated, stated, official document image provided in LOTE, or
  - Finalized
  - And a translation status
    - official document image provided in English, or
    - approved translation of official document provided

19. All applications will have a status object
    - That will contain
      - Current RHD staff supervisor
      - Previous staff supervisor
      - Is awaiting information (yes/no – info in correspondence)
      - Specific Application Status
        - Specific Status is
          - Terminated/Rejected – application is canceled
          - Unfinished – the application is unfinished
          - Proposed – the application is finished and proposed
          - Flagged – staff have flagged as interested
          - Partially assigned – some staff have agreed to supervise
          - Assigned – staff have agreed to supervise
          - Formal pending – all details are in order, formal process starts
          - Approved – formal application is approved
      - Decisions made
        - Made by
        - Conclusion reached
        - Comment
        - Date made
20. On an application status, flagged staff members are notified
21. The decision process will determine the applications status, application status starts at unfinished. Progress through the following decisions:

Question	**Decision**	**action – Application status**
Has the application been submitted	Yes	If unfinished change to Proposed
	No	(keep current status)
Have some staff flagged (0 to less than two)	Yes	If proposed change to Flagged
	No	(keep current status)
Is this an application worth pursuing (maybe no flags, no potential supervisors over six months)?	Yes	(keep current status)
	No	Terminated
Has one supervisor?	Yes	If Flagged/proposed change to partially assigned

	No	(keep current status)
There are two supervisors?	Yes	If Flagged/proposed/ partially assigned change to Assigned
	No	(keep current status)
Is application flagged and or has one supervisor and is aimed to commence in less than 2 months	Yes	Inform flaggees the application is about to expire
	No	(keep current status)
If requested additional information, how long has the applicant not responded to requests	>1 month	(keep current status)
	1 month	Send reminder
If requested additional information, how long has the applicant not responded to requests Has the applicant provided all the information we require?	3 months	Send reminder
	6 months	Terminate application
	Yes	Change application status to Start formal pending. Send email to being RHD Research formal application
	No	Issue request for information (from proforma, add correspondence entry, Set information requested to true)
Has enough information been presented to start a formal application?	Yes	Start formal application
	No	(keep current status)
Has the formal application been completed?	Yes	Mark application as approved, send confirmation email + official letter
	No	(keep current status)

The decisions that are still required are:

- What constitutes a minimal form that can be proposed (up for flagging and potential supervisors?)
- The leeway between proposed start date and when the application is to be rejected/

Data types:

- All data stored in the database will be hyperlinks, varChars, integers, Booleans, dates and SmallText
- The exact types each data will have will be detailed in the conceptual, logical and physical diagrams.
- The database will have to maintain links to attached document files, e.g. PDF and image files.

1.3.1 Academic Staff Functional requirements

1. Inherits all the functional requirements of the Professional Staff view
2. Academic staff will be able to flag applications as interested, including a link between staff and application
3. Academic staff will be able to flag applicants as interested, including a link between staff and application
4. Each academic member of staff may store information about fields of research they may be interested in supervising
    4.1. A list of Field of Research codes
    4.2. A list of user-defined keywords
5. Generate a report of summary of all applications with matching fields of research
    5.1. Optionally filtered to only new applications since a specified date
6. Receive notifications of new applications that have matching research field keywords.

1.3.2 RHD Co-ordination Staff Functional Requirements

1. Inherit all the functions of the Professional and Academic Staff views
2. Report statistics on the number of applications being actively managed and their status
3. Report statistics on the speed of processing RHD applications, to help decide if the system is meeting performance requirements

1.4 Change log

a) Re-arranged layout and flow of this document to make it more specific to database design.
b) Added preliminary discussion on user views, after feedback

2 Conceptual Design

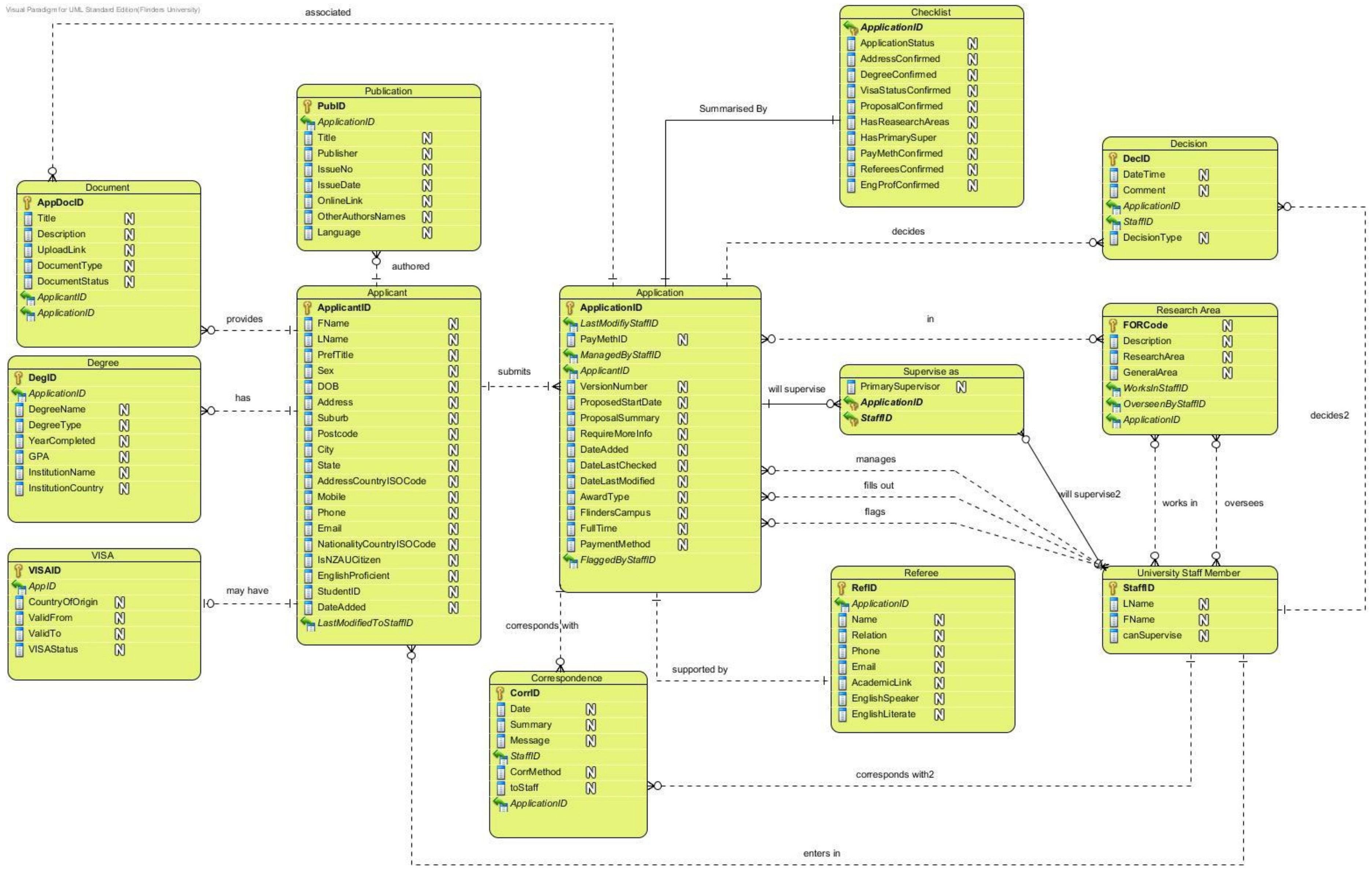


Figure 2 Conceptual E-R diagram

2.1 Identified entity types

2.1.1 Strong Entities

Entity Name	**Aliases**	**Description**	**Occurrences (select multiplicities)**	**Entity #**
Applicant	-	Holds the applicant specific details	Can be zero at the start but is expected to be many during use	1
Document	-	Links to any relevant documents along with descriptions and types.	Zero or more for each applicant, specific to the Applicant.	2
Checklist	-	Allows users to record the received and outstanding application information.	Always exactly one for each application.	3
Degree	-	Any Degrees already held by the applicant	One or more for each applicant, specific to the Applicant.	4
Application	-	Holds the application details	one or more for every Applicant	5
Publication	-	Publications that the applicant has authored	Zero or more of their publications can be added for each Applicant	7
Referee	-	A referee for the applicant	One or more for each Applicant	8
Research Area	-	The areas of a research an application or staff member is in.	a set number of predefined research areas, reusable by Staff and Applicants	9
University Staff Member	-	Supervisor or RHD Staff Member	Includes both of the above	12
Visa	-	The applicants visa details	Zero or one for each Applicant (as required)	13

2.1.2 Weak Entities

Entity Name	**Aliases**	**Description**	**Occurrences (select multiplicities)**	**Entity #**
Correspondence	Message	Correspondence between the Applicant and University Staff Member	Zero or more for every application	14
Decision	Change Log	The decision and comment made for an application by a RHD staff member	Zero or more for every application	15
SuperviseAs	Supervisor type	Supervise as a primary or secondary supervisor	One for ever supervisor	16

2.2 Identify relationship types

Entity Name 1	**multiplicity**	**Relationship Name**	**multiplicity**	**Entity Name 2**	**Rel#**
Applicant	1	provides	0..*	Document	1.1
	1	Submits	1..*	Application	1.2
	1	holds	1..*	Degree	1.3
	1	has authored	0..*	Publication	1.4
	1	May Require	0..1	Visa	1.5
Application	0..*	In	1..*	ResearchArea	2.1
	0..*	Prefers	1..3	Supervise as	2.2
	1	Summarized By	1	Checklist	2.3
	1	Supported By	0..5	Referee	2.4
Document	0..*	associated	0..1	Application	5.1
	0..*	provides	1..1	Applicant	5.2
RHDStaffMember	1	Decides	0..*	Application	10.1
	0..*	Oversees	0..*	ResearchArea	10.2
	1	Manages	0..*	Application	10.3
Supervise as	0..*	Flags	0..*	Application	11.1
	0..*	Will supervise	0..*	Application	11.2
	0..*	Works in	1..*	ResearchArea	11.3
UniversityStaffMember	0..*	Corresponds with	0..*	Applicant	12.1
	1	enters in	0..*	Applicant	12.2
	1	Fills out	0..*	Application	12.3

2.3 Identify and associate attributes with entity or relationship types

2.4 Attributes by Entity

Entity Name	Attribute Name	Description	Data type and Length*	Allow nulls	Multi - valued	attri#
Applicant	fName	First name	VARCHAR(50)	N	N	1.1
	lName	Last name	VARCHAR(50)	Y	N	1.2
	prefTitle	Title Mr, Mrs, Miss, Dr. *	CHAR(10)	Y	N	1.3
	sex	The sex of the applicant	TINYINT(1)	N	N	1.4
	DOB	Date of birth	DATETIME	Y	N	1.5
	address	Residence number and street of residence	VARCHAR(254)	Y	Y	1.6
	suburb	The suburb of residence	VARCHAR(100)	Y	N	1.7
	postcode	The postcode of residence	MEDIUMINT	Y	N	1.8
	city	The city or town of residence	VARCHAR(50)	Y	N	1.9
	state	The State of residence	VARCHAR(50)	Y	N	1.10
	country	The country of residence	CHAR(2)**	Y	N	1.11
	mobile	Mobile phone number	VARCHAR (50)	Y	N	1.12
	phone	Landline phone number	VARCHAR (50)	Y	N	1.14
	email	The email address of the applicant	VARCHAR(100)	N	N	1.15
	nationality	The nationality of the applicant	CHAR(2)**	Y	N	1.16
	isNZAUCitizen	Is a new Zealand or Australian citizen – a check to see if visa information is required ***	TINYINT(1)	Y	N	1.17
	englishProficient	English ability	TINYINT	Y	N	1.18
	studentID	The flinders university student id if they are or have been enrolled at flinders university	UNSIGNED MEDIUMINT	Y	N	1.19
	dateAdded	The date the applicant was added to the system	DATETIME	N	N	1.20
Application	proposedStartDate	The date the applicant prefers to start the RHD (Entered as 1/1/## for S1 and 1/7/## for S2)	DATETIME	Y	N	2.1
	proposalSummary	What the proposal is about	VARCHAR(2000)	Y	N	2.2
	dateAdded	The date the application was submitted/saved	DATETIME	N	N	2.3
	dateLastChecked	The date the application was last checked	DATETIME	N	N	2.4
	dateLastModified	the date the application was last modified	DATETIME	N	N	2.5
	paymentMethod	The Type of payment method	VARCHAR(50)	Y	N	2.6
	awardType	An award name	VARCHAR(50)	N	N	2.7
	FlindersCampus	If the study will be on site	Boolean	Y	N	2.8
	FullTime	If ti will be full-time study	Boolean	Y	N	2.9
Checklist	applicationStatus	The status of the application	VARCHAR(20)	N	N	3.1
	addressConfirmed	All contact details appear valid	TINYINT(1)	N	N	3.2
	degreeConfirmed	The degree is a recognised degree of the institution	TINYINT(1)	N	N	3.3
	visaConfirmed	The visa status is backed by an official document	TINYINT(1)	N	N	3.4
	proposalConfirmed	The proposal is contains appropriate detail	TINYINT(1)	N	N	3.5

2.4 Attributes by Entity

Entity Name	Attribute Name	Description	Data type and Length*	Allow nulls	Multi - valued	attri#
	engProfConfirmed	The applicant has some level of English literacy	TINYINT(1)	N	N	3.6
	hasResearchAreas	Has nominated research areas relevant to the proposal	TINYINT(1)	N	N	3.7
	hasPrimarySuper	Has the required number of supervisors	TINYINT(1)	N	N	3.8
	payMethConfirmed	The payment method is backed by an official document	TINYINT(1)	N	N	3.9
	refereesConfirmed	The referees details appear to be correct	TINYINT(1)	N	N	3.10
Degree	name	The title of the degree	VARCHAR(100)	N	N	4.1
	type	The type of the degree - Could add specific types	VARCHAR(100)	N	N	4.2
	yearCompleted	The year the degree was completed or will be completed	DATETIME	Y	N	4.3
	GPA	The GPA of the degree	VARCHAR(5)	Y	N	4.4
	institutionName	The name of the institution	VARCHAR(100)	Y	N	4.5
	institutionCountry	The country the institution is located in	CHAR(2)**	Y	N	4.6
Document^	title	The title of the document	VARCHAR(254)	N	N	5.1
	description	A specific summary related to the document i.e. valid till 2015 etc.	VARCHAR(2000)	Y	N	5.2
	uploadLink	An link to a version uploaded and stored on the university servers	VARCHAR(254)	N	N	5.3
	docType	The type of document	VARCHAR(50)	N	N	5.4
	docStatus	Official and translation status of a document associated to an Applicant	VARCHAR(50)	N	N	5.5
Publication	title	The title of the publication	VARCHAR(254)	N	N	7.1
	abstract	A abstract/description of the publication	VARCHAR(2000)	Y	N	7.2
	publication	The journal/magazine publisher	VARCHAR(254	N	Y	7.3
	issueNo	The issue/edition number of the publication	MEDIUMINT	Y	N	7.4
	issueDate	The date the publication was issued	DATETIME	N	N	7.5
	onlineLink	An online link to the publication	VARCHAR(254)	Y	N	7.6
	otherAuthorsNames	Other authors of the publication	VARCHAR(254)	Y	Y	7.7
	language	language of the publication	VARCHAR(50)	Y	N	7.8
Referee	name	The full name of the referee	VARCHAR(100)	N	N	8.1
	relation	The referees relation to the applicant	VARCHAR(100)	N	N	8.2
	phone	The referees phone number	VARCHAR(50)	Y	N	8.3
	email	The referees email address	VARCHAR(100)	N	N	8.4

2.4 Attributes by Entity

Entity Name	Attribute Name	Description	Data type and Length*	Allow nulls	Multi - valued	attri#
	profession	The referees profession	VARCHAR(254)	N	N	8.5
	academicLink	The referees professional page (linked in or university)	VARCHAR(254)	Y	N	8.6
	englishSpeaker	If the Referee can speak English	TINYINT(1)	Y	N	8.7
	englishLiterate	If the Referee can read and write in English	TINYINT(1)	Y	N	8.8
ResearchArea	FORCode	The Australian Field Of Research (FOR) code	SMALLINT	N	N	9.1
	Description	A small text description of the FOR i.e. 2201	VARCHAR(2000)	N	N	9.2
	researchArea	The FOR title; i.e. Applied Ethics	VARCHAR(50)	N	N	9.3
	generalArea	The general area of the FOR	VARCHAR(50)	N	N	9.4
Supervise as	-	-	-	-	-	11
UniversityStaff Member	staffID	The flinders uni staff id number	MEDIUMINT	N	N	12.1
	fName	The first name of the staff member	VARCHAR(50)	N	N	12.2
	lName	The last name of the staff member	VARCHAR(50)	N	N	12.3
	canSupervise	If the staff member can supervise RHD candidates	Boolean	N	N	12.4
Visa	validFrom	When the visa is valid from	DATETIME	Y	N	13.1
	validTo	When the visa is valid to	DATETIME	Y	N	13.2
	VisaStatus	Approved unapproved, submitted, waiting etc.	VARCHAR(50)	N	N	13.3

2.5 Attributes types by Association Entity

Entity Name	Attribute Name	Description	Data type and Length*	Allow nulls	Multi - valued	attri#
Correspondence	date	The date the correspondence was made/received	DATETIME	N	N	14.1
	summary	A small summary of the Correspondence	VARCHAR(1000)	Y	N	14.2
	message	The actual message contained in the correspondence	VARCHAR(1000)	N	N	14.3
	corrMeth	The method of correspondence.	VARCHAR(50)	N	N	14.4
	toStaff	The direction of the correpondance	TINYINT(1)	N	N	14.5
Decision	date	The date the decision was made on	DATETIME	N	N	15.1
	comment	Extra information about the decision	VARCHAR(1000)	Y	N	15.2
	decType	The type of decision	VARCHAR(50)	N	N	15.3
SuperviseAs^^	primarySupervisor	True will supervise as primary supervisor else as a secondary supervisor	TINYINT(1)	N	N	16.1

Related information

* types not specifically set
** uses the ISO 3166-1 alpha-2 code
*** as per the Australian High Commission New Zealand - http://www.newzealand.embassy.gov.au/wltn/study.html
^ a degree, publication, visa or other document may include
^^ primary supervisors are decided on a first come first serve basis

Field lengths are based on the approximation that 500 chars is approximately 100 words
2000~400 words

2.6 Determined Domain attribute domains

Entity Name	Attribute Name	Domain (includes validation information)	Default value	attri#
Applicant	prefTitle	-	-	1.3
	Sex	Male/Female	-	1.4
	DOB	>16 years old	-	1.5
	country	ISO 3166-1 alpha-2 code	-	1.11
	email	Is a valid Email address	-	1.15
	Nationality	ISO 3166-1 alpha-2 code	-	1.16
	isNZAUCitizen	y/n	-	1.17
	englishProficient	(~0-120) TOEFL Score	-	1.18
	studentID	>1,000,000 and <2,500,000	-	1.19
	dateAdded	Is a date	CURDATE()	1.30
Application	proposedStartDate	> CURDATE() and CURDATE()+5years<	-	2.1
	dateAdded	-	CURDATE()	2.3
	dateChecked	-	CURDATE()	2.4
	dateLastModified	Always CURDATE() on Application or its relationships change	CURDATE()	2.5
	paymentMethod	- student loan - savings - government assistance (AU) - government assistance (Foreign government) - scholarship	-	2.6
	awardType	- Master of Arts - Master of Biotechnology - Master of Business - Master of Clinical Education - Master of Clinical Rehabilitation - Master of Engineering - Master of Laws - Master of Science - Master of Surgery - Master of Theology - Doctor of Philosophy (PhD) in all disciplines - Doctor of Philosophy (Clinical Psychology) – includes some coursework. - Doctor of Education - Doctor of Public Health Cotutelle Doctorate – (Flinders University and an international higher education institution)	-	2.7
	FlindersCampus	- T/F	-	2.8
	FullTime	- T/F	-	2.9
Checklist	applicationStatus	- Rejected – application is rejected - not eligible – applicant is found to be unable to apply - Unfinished – the application still requires information/unfinished - Proposed – the application is finished and proposed - Flagged – staff have flagged as interested - Partially assigned – some staff have agreed to supervise - Assigned – staff have agreed to supervise - Formal pending – all details are in order, formal process starts - Approved – formal application is approved	Unfinished	3.1
	addressConfirmed	Y/N	N	3.2
	degreeConfirmed	Y/N	N	3.3
	visaConfirmed	Y/N	N	3.4
	proposalConfirmed	Y/N	N	3.5
	engProfConfirmed	Y/N	N	3.6
	hasResearchAreas	Y/N	N	3.7
	hasPrimarySuper	Y/N	N	3.8
	payMethConfirmed	Y/N	N	3.9
	refereesConfirmed	Y/N	N	3.10
Degree	type	- Certificate - Associate Degrees - Diploma - Postgraduate diploma - Bachelor Degree - Bachelor(honours) - Graduate Diploma - Graduate Certificate - Masters - Graduate Entry Masters - Professional Masters	-	4.2

		- PHD		
	yearCompleted	>CURDATE-2YEARS	-	4.3
	GPA	>0 and <10	-	4.4
	institutionCountry	ISO 3166-1 alpha-2 code	-	4.6
Document	uploadLink	Is a URL	-	5.0
	docType	- Paper - Academic transcript - Proposal - Proof of financial ability - Visa application acceptation paper - other	-	5.4
	docStatus	- Official - Unofficial - Official translation - Unofficial translation - N/A	-	5.5
Publication	issueDate	>CURDATE-50YEARS and <CURDATE+1	-	7.5
	onlineLink	isURL	-	7.6
Referee	phone	isPhone	-	8.3
	email	isEmail	-	8.4
	academicLink	Is html page	-	8.6
	englishSpeaker	Y/N	-	8.7
	englishLiterate	Y/N	-	8.8
ResearchArea	FORCode	FORCode	-	9.1
	Description	-	-	9.2
	researchArea	FORCode tilte	-	9.3
	generalArea	- DIVISION 01 MATHEMATICAL SCIENCES - DIVISION 04 EARTH SCIENCES - DIVISION 05 ENVIRONMENTAL SCIENCES - DIVISION 08 INFORMATION AND COMPUTING SCIENCES - DIVISION 09 ENGINEERING - DIVISION 10 TECHNOLOGY	-	9.4
VISA	countryOfOrigin	ISO 3166-1 alpha-2 codes	-	13.1
	VISAStatus	- student visa - unknown	unknown	13.2

Attribute Entity Domains

Correspondence	CorrMethod	- Phone - Email - Skype - Mail	-	14.1
	toStaff	Y/N	-	14.5
Decision	Type	- Status change – Application status change (status to from added automatically by - Info request – a request has been made for Information - requested - Application handed over to	-	15.1

- Not null no default
* Possible but will not be implemented

2.7 Determined candidate, primary, and alternate key attributes

As we want to support use with only minimal partial data. Hence, almost all attributes accept NULL. As nullable fields cannot participate in

Entity Name	Primary Key	Alternative Key	Entity#
Applicant	applicantID	emailAddress lName fName, Phone lName fName streetAddress lName fName,	1
Document	AppDocD	-	2
Application	applicationID	-	3
Checklist	applicantID	-	4
Degree	degreeID	-	5
GeneralStaffMember	staffID	-	6
Publication	pubID	-	7
Referee	refID	-	8
ResearchArea	FORCode	-	9
RHDStaffMember	staffID	-	10
Supervise as	staffID	fName + Research Areas	11
UniversityStaffMember	staffID	-	12
Visa	visaID	-	13
Correspondence	corrID	Date & Staff ID & applicantID	14
Decision	decID	-	15
SuperViseAs	staffID&applicaitonID	-	16

Most relations do not have any alterative primary keys i.e. a person can have the same checklist state, hold the same degrees, specify the same supervisors and while unlikely published the same publications, have the same referees and have similar visa details.

It should be noted that Supervisor, General Staff Member and RHD staff Member all have the same primary key as they will all be merged into University Staff Member with a discriminated added to state the type as none of these specialised classes have any specific data types.

2.8 Use enhanced modelling concepts

We investigated the use of generalisation relations between different staff member types. It was found that this approach did not yield much simplification, as there are few attributes either shared or distinct between the staff types.

We investigated the use of generalisation relation from Application Document to Publication, Degree, Visa, Application or Payment Method. This approach was rejected as we found there were too many distinct attributes among the specialised entities.

2.9 Check model for errors and redundancy

(1) Re-examine one-to-one (1:1) relationships;
All entity objects identified are unique or form part of a generalisation specialisation set
The checklist 1-1 relationship is preserved to make it separate to the application it is related to

(2) Remove redundant relationships;
Applicant provides ApplicationDocument left in to account for other documents
No redundant relationships were found as each relationship in the diagram reflects a unique database transaction

(3) Consider time dimension.
Many relationships multiplicities are "zero to"s to signify that details for a degree, VISA etcs are not available or cannot be entered at the current time.
These are then checked through not nulls and noted as being valid for the application (not the DB) in the checklist.

(4) Check for fan traps and chasm traps.
Fan Trap
Where a model represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous.
Chasm Trap
Where a model suggests the existence of a relationship between entity types, but pathway does not exist between certain entity occurrences.

2.10 Validate conceptual model against user transactions

Identify transaction pathways and check that they will work.
The following user pathways were tested:

Table 1 User transactions for all staff, including professional staff

a)	Look up applicant + publications + degrees + visa Status + Associated documents by applicant name
b)	Look up applicant's applications by applicant name
c)	Look up applicant's applications by applicant email
d)	Look up incomplete applications
e)	Look up all correspondences relevant to the application
f)	Insert Applicant
g)	Who edited the Applicant/Application last
h)	Create new applicant and associated application records
i)	Look up an existing application and attach a new standard type document to an application
j)	Look up an existing application and attached a new exceptional type document to an application
k)	Look up an existing application and list outstanding information (checklist).
l)	Elevate the certainty status flag of a piece of information relating to an existing application
m)	Retrieve all on-going applications for which the user has made the most recent correspondence
n)	Record making a decision about an application
o)	Update the status of an application

Table 2 User transactions for academic staff

p)	Look up, add to, and delete from own current research areas
q)	Search for all applications in certain research areas that have been added since a certain time
r)	Flag interest in an application

Table 3 User transactions for RHD co-ordinations staff

s)	Who edited the Applicant/Application last
t)	Retrieve all ongoing applications
u)	List all ongoing applications that do not have a primary supervisor allocated.

Note that going through transaction f) forced changes to the conceptual design; these are noted in the Change Log section 2.13 item a).

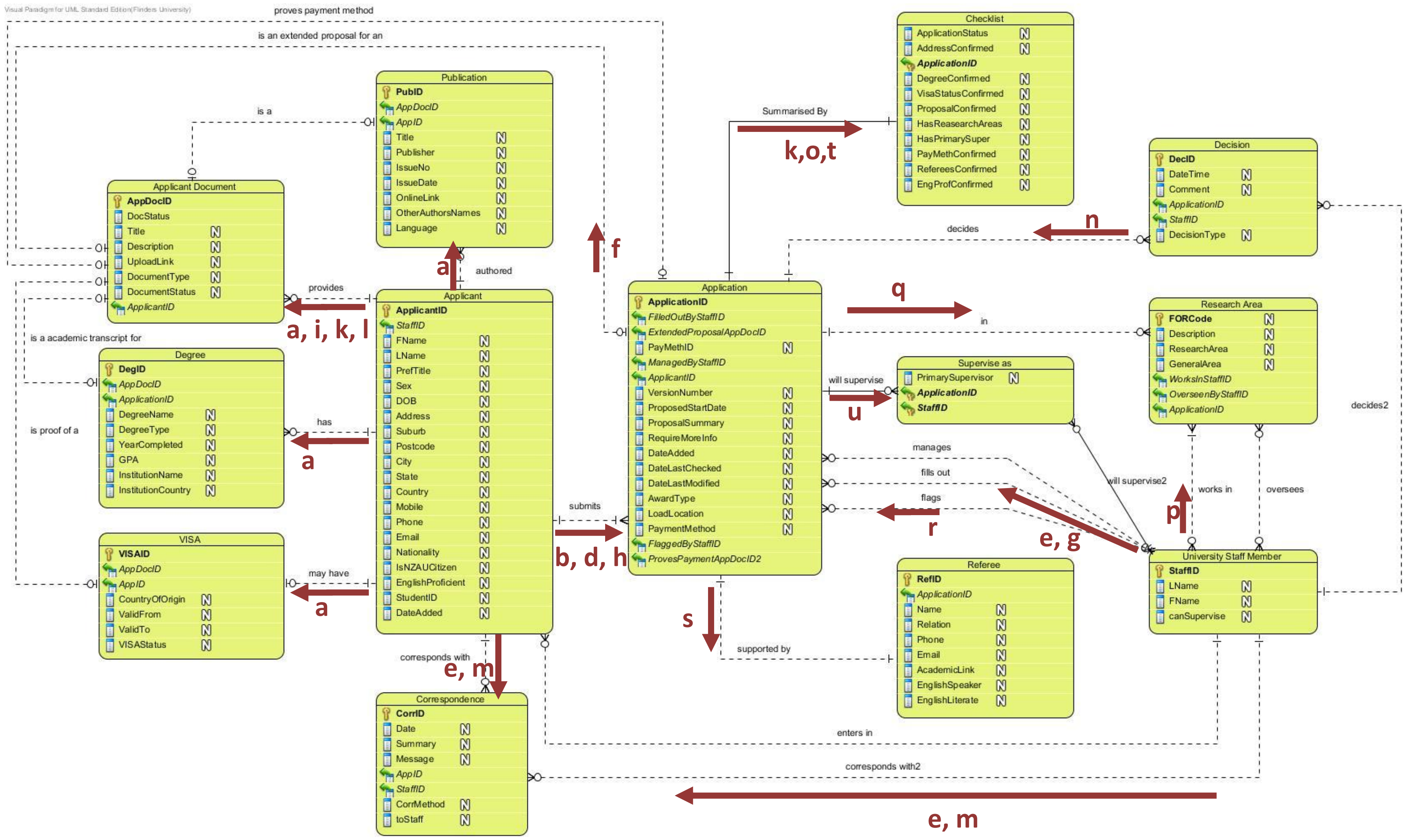

Visual Paradigm for UML Standard Edition(Flinders University)
proves payment method
is an extended proposal for an
is a
Publication
PubID
AppDocID
AppID
Title
Publisher
IssueNo
IssueDate
OnlineLink
OtherAuthorsNames
Language
Applicant Document
AppDocID
DocStatus
Title
Description
UploadLink
DocumentType
DocumentStatus
ApplicantID
provides
authored
a
a, i, k, l
f
is a academic transcript for
Degree
DegID
AppDocID
ApplicationID
DegreeName
DegreeType
YearCompleted
GPA
InstitutionName
InstitutionCountry
has
a
is proof of a
VISA
VISAID
AppDocID
AppID
CountryOfOrigin
ValidFrom
ValidTo
VISAStatus
may have
a
Applicant
ApplicantID
StaffID
FName
LName
PrefTitle
Sex
DOB
Address
Suburb
PostCode
City
State
Country
Mobile
Phone
Email
Nationality
IsNZAUCitizen
EnglishProficient
StudentID
DateAdded
submits
b, d, h
corresponds with
e, m
Correspondence
CorrID
Date
Summary
Message
AppID
StaffID
CorrMethod
toStaff
Application
ApplicationID
FilledOutByStaffID
ExtendedProposalAppDocID
PayMethID
ManagedByStaffID
ApplicantID
VersionNumber
ProposedStartDate
ProposalSummary
RequireMoreInfo
DateAdded
DateLastChecked
DateLastModified
AwardType
LoadLocation
PaymentMethod
FlaggedByStaffID
ProvesPaymentAppDocID2
Summarised By
k,o,t
Checklist
ApplicationStatus
AddressConfirmed
ApplicationID
DegreeConfirmed
VisaStatusConfirmed
ProposalConfirmed
HasReasearchAreas
HasPrimarySuper
PayMethConfirmed
RefereesConfirmed
EngProfConfirmed
Decision
DecID
DateTime
Comment
ApplicationID
StaffID
DecisionType
decides
n
q
in
Research Area
FORCode
Description
ResearchArea
GeneralArea
WorksInStaffID
OverseenByStaffID
ApplicationID
will supervise
u
Supervise as
PrimarySupervisor
ApplicationID
StaffID
decides2
manages
fills out
flags
r
e, g
will supervise2
works in
oversees
p
University Staff Member
StaffID
LName
FName
canSupervise
s
supported by
Referee
RefID
ApplicationID
Name
Relation
Phone
Email
AcademicLink
EnglishSpeaker
EnglishLiterate
enters in
corresponds with2
e, m

Figure 3

2.11 Develop draft test plan

We will generate a test data set to represent a baseline state of a running system. This includes a way to automatically load this baseline data into database system. This data set will need to be realistic, to ensure we are testing the system in a similar way in which it will be used. It will also have to be comprehensive to ensure that it tests integrity constraints.

We will also generate scripts that perform additional create, retrieve, update and delete operations that simulate a set of realistic user transactions, as listed in section 2.10. These scripts will include details of the expected result of each operation, to enable logging success/failure and time taken of each test.

As changes are made to the implementation of the system, we will compare the test logs before and after changes to ensure the system performance is improving overall.

As problems are discovered in the system implementation, ensure that the test suites are updated to include these newly discovered problems, to prevent reintroducing these errors.

We will evaluate for use the unit testing framework STK/Unit (Picchiarelli and Razzoli 2013). If this product works as claimed, using it will simplify constructing and executing tests, and also evaluating the system performance against those tests, and tracking that performance over time.

The functions to be tested include that all the user transactions are supported:

#	pathway
a)	Look up applicant + publications + degrees + visa Status + Associated documents by applicant name
	Tested by five separate queries, where
b)	Look up applicant's applications by applicant name
c)	Look up applicant's applications by applicant email
d)	Look up incomplete applications
e)	Look up all correspondences relevant to the application
f)	Insert Applicant
g)	Who edited the Applicant/Application last
h)	Create new applicant and associated application records
i)	Look up an existing application and attach a new standard type document to an application
j)	Look up an existing application and attached a new exceptional type document to an application
k)	Look up an existing application and list outstanding information (checklist).
l)	Elevate the certainty status flag of a piece of information relating to an existing application
m)	Retrieve all on-going applications for which the user has made the most recent correspondence
n)	Record making a decision about an application
o)	Update the status of an application

Academic Staff

#	pathway
p)	Look up, add to, and delete from own current research areas

q)	Search for all applications in certain research areas that have been added since a certain time
r)	Flag interest in an application

RHD Co-ordination Staff

#	pathway
s)	Who edited the Applicant/Application last
t)	Retrieve all ongoing applications
u)	List all applications waiting for supervisor agreement

2.12 Review conceptual data model

The conceptual model has been reviewed against the SRS, the original project description and reviewed by an end-user stakeholder (tutor). It was found that conceptual model presented here in will meet the data requirements of RHD admissions office.

The decisions made and new terminology in this document have been appended to the requirements specification.

2.13 Change Log

a) Document type, Document Status, Application Status, Correspondence Method, Payment Method and Decision types all moved to separate entities of know finite tuples
b) Application's Fulltime and internal attributes combined and moved to Study Load and Location
c) dateLastModifed and dateLastChecked added to application, these will also be updated in the latest application version if an applicant is updated at the same time as requiresMoreInfo is checked
d) fills out and enters in relationships added to University Staff Member
e) Supervise as added moved supervise relationship.
f) country added to Conceptual diagram but no added here
g) Removed Document type, Document Status, Application Status, Correspondence Method, Payment Method and Decision types entities, replaced with attributes. The document was then updated to reflect this change.
h) Added user transaction u) after feedback provided.
i) Updated the model and diagram to reflect later simplifications.
j) Updated enhanced modelling concepts section to reflect simplifications.

2.14 Other considerations and Decisions to be made

Question	**affects**
Does an applicant always say what research they want to do?	Submits application multiplicity
Is there a GPA conversion method available	

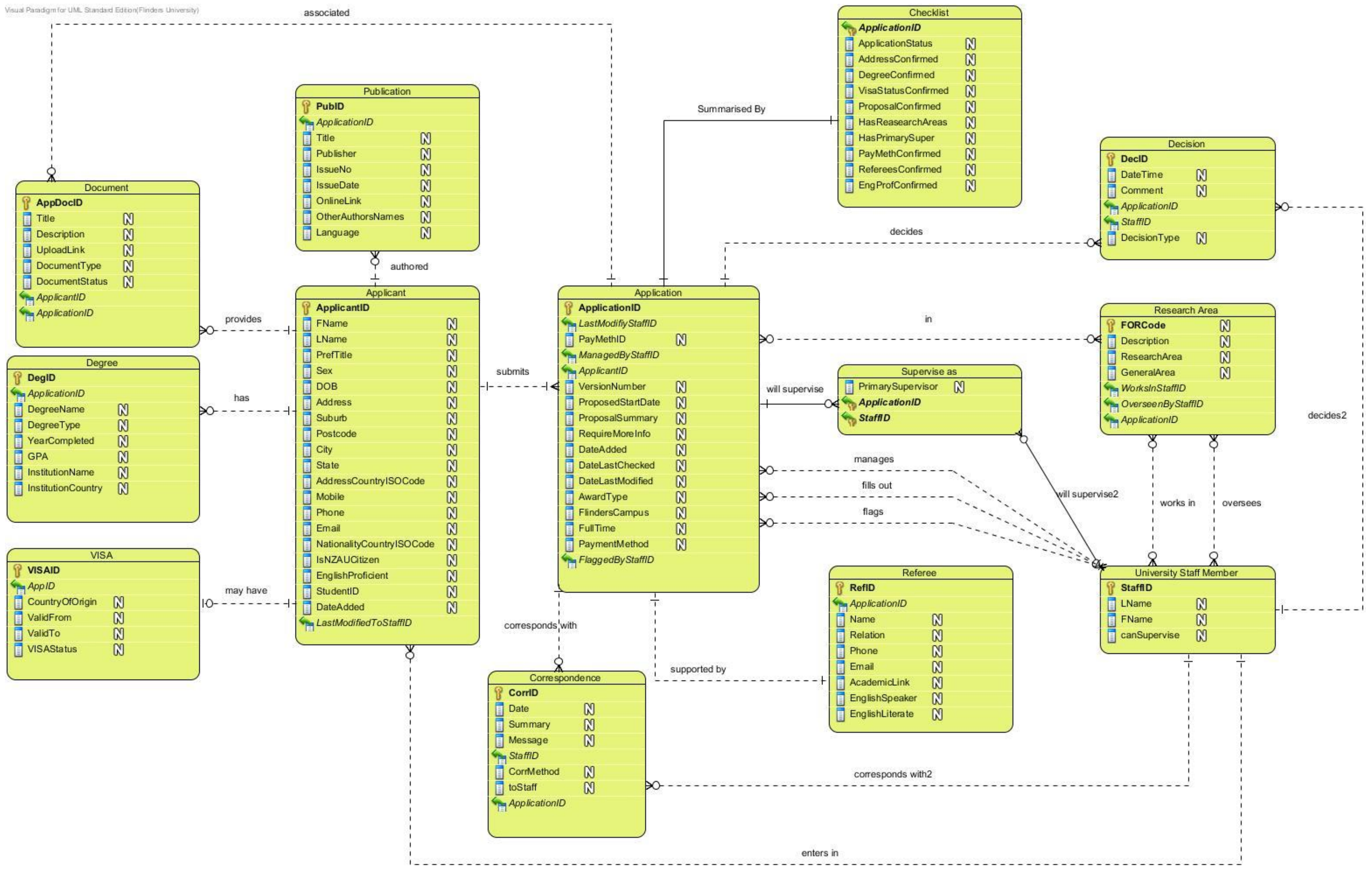
Assumptions

- Applicant will always have enough information for an Application (proposal summary, start date, award they seek and load and location)
- The application will run the minimum submission checklist (not the checklist entity which is for user use only) after each update and: update the most recent applications; if more information is required; and update the dateLastModified.

3 Logical Design

3.1 Derive relations

3.1.1 Conceptual E-R diagram



3.1.2 Strong Entity types

Applicant (ApplicantID, FName, LName, PrefTitle, Sex, DOB, StreetAddress, Suburb, Postcode, City, State, Country, Mobile, Phone, Email, Nationality, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID) **Primary key** ApplicantID
Application (ApplicationID, ProposedStartDate, ProposalSummary, RequireMoreInfo, DateAdded, DateLastChecked, DateLastModified, AwardType, flindersCampus, fullTime, PaymentMethod) **Primary key** ApplicationID
Correspondence (CorrID, Date, Summary, Message, CorrMeth, toStaff) **Primary key** CorrID
Decision (DecID, DateTime, Comment, DecisionType) **Primary key** DecID
Degree (DegID, DegreeName, DegreeType, YearCompleted, GPA, InstitutionName, InstituitonCountry) **Primary key** DegID
Document (DocID, UploadLink, DocStatus, DocumentType, Title, Description) **Primary key** UploadLink
Publication (PubID, ApplicantID, Title, Publication, IssueNo, IssueDate, OnlineLink, OtherAuthorsNames, Language) **Primary key** PubID
Referee (RefID, Name, Relation, Phone, Email, AcademicLink, EnglishSpeaker, EnglishLiterate) **Primary key** RefID
Research Area (FORCode, Description, ResearchArea, GeneralArea) **Primary key** FORCode
University Staff Member (StaffID, FName, LName, canSupervise) **Primary key** StaffID
Visa (VisaID, VISAStatus, CountryOfOrigin, ValidFrom, ValidTo) **Primary key** VisaID

3.1.3 Weak Entity types

Checklist (ApplicationStatus, AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchArea, HasPrimarySuper, PayMethConfirmed, RefrereesConfirmed, EngProfConfirmed) **Primary key** None (at this stage)
Supervise as (PrimarySupervisor)* **Primary key** None (at this stage)

* This entity was created during the conceptual phase to remove a relationship attribute

Very few natural alternate keys were found as many attributes are nullable due the limited information received at the time of initial user entry.

3.1.4 One-to-many binary relationships

Relationship name	Identifying	Parent relation	Parent multiplicity	Foreign Keys	Child relation	Child multiplicity
associated	No	Application	0..1	ApplicationID	Document	0..*
authored	No	Applicant	1..1	ApplicantID	Publication	0..*
corresponds with	No	Application	1..1	ApplicationID	Correspondence	0..*
corresponds with2	No	University Staff Member	1..1	StaffID	Correspondence	0..*
decides	No	Application	1..1	ApplicationID	Decision	0..*
decides2	No	University Staff Member	1..1	StaffID	Decision	0..*
has	Yes	Applicant	1..1	ApplicantID	Degree	0..*
last to modify		University Staff Member	1..1	StaffID → LastToModifyStaffID	Applicant	0..*
last to update	No	University Staff Member	1..1	StaffID →LastToModifyStaffID	Application	0..*
manages	No	University Staff Member	0..1	StaffID → ManagedByStaffID	Application	0..*
provides	No	Applicant	1..1	ApplicantID	Document	0..*
submits	Yes	Applicant	1..1	ApplicantID	Application	0..*
supported by	No	Application	1..1	ApplicationID	Referee	1
will supervise	Yes	Application	1..1	ApplicationID	Supervise as	0..*
will supervise2	Yes	University Staff Member	1..1	StaffID	Supervise as	0..*

Included in the relations of section 1.2 this becomes

Applicant (ApplicantID, FName, LName, PrefTitle, Sex, DOB, StreetAddress, Suburb, Postcode, City, State, AddressCountryISOCode, Mobile, Phone, Email, NationalityCountryISOCode, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID)
Primary key ApplicantID
Foreign key LastToModifyStaffID **references** University Staff Member(StaffID)

Application (ApplicationID, applicationStatus, AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchAreas, HasPrimarySuper, PayMethConfirmed, EngProfConfirmed, RefereesConfirmed, LastToModifyStaffID, ProposedStartDate, ProposalSummary, RequireMoreInfo, DateAdded, DateLastChecked, DateLastModified, awardType, flindersCampus, fullTime, paymentMethod)
Primary key ApplicationID
Foreign key ApplicantID **references** Application(ApplicationID)
Foreign key LastToModifyStaffID **references** University Staff Member(StaffID)
Foreign key ManagedByStaffID **references** University Staff Member(StaffID)

Application_Research Area (ApplicationID, FORCode)
Primary key ApplicationID, FORCode
Foreign key ApplicantID **references** Application(ApplicationID)

Correspondence (CorrID, Date, Summary, Message, ApplicantID, StaffID, CorrMeth, toStaff
Primary key CorrID
Foreign key ApplicantID **references** Applicant(ApplicantID)
Foreign key StaffID **references** UniversityStaffMember(StaffID)

Decision (DecID, Date, Comment, ApplicationID, StaffID, decType)
Primary key DecID
Foreign key ApplicationID **references** Application(ApplicationID)
Foreign key StaffID **references** UniversityStaffMember(StaffID)

Degree (DegID, ApplicantID, Name, Type, YearCompleted, GPA, InstitutionName, InstitutionCountryISOCode)
Primary key DegID
Foreign key ApplicantID **references** Applicant(ApplicantID)

Document (DocID, UploadLink, Title, Description, DocType, DocStat, ApplicantID, ApplicationID)
Primary key UploadLink
Foreign key ApplicantID **references** Applicant(ApplicantID)
Foreign key ApplicationID **references** Application(ApplicationID)

Publication (PubID, ApplicantID, Title, Publication, IssueNo, IssueDate, OnlineLink, OtherAuthorsNames, Language)
Primary key PubID
Foreign key ApplicantID **references** Applicant(ApplicantID)

Referee (RefID, ApplicationID, Name, Relation, Phone, Email, AcademicLink, EnglishSpeaker, EnglishLiterate)
Primary key RefID
Foreign key ApplicationID **references** Application(ApplicationID)

Research Area (FORCode, Description, ResearchArea, GeneralArea) UNCHANGED
Primary key FORCode

Supervise as (StaffID, ApplicationID, PrimarySupervisor)
Primary key StaffID, ApplicationID
Foreign key ApplicationID **references** Application(ApplicationID)
Foreign key StaffID **references** University Staff Member(StaffID)

University Staff Member (StaffID, FName, LName, canSupervise) UNCHANGED
Primary key StaffID

Visa (VisaID, ApplicantID, OriginCountryISOCode, VISAStatus, ValidFrom, ValidTo) UNCHANGED
Primary key VisaID

3.1.5 One-to-one binary relationships

3.1.5.1 Mandatory on both sides

For the **Application *summarised by* Checklist** relationship, we bring all the attributes of **Checklist** into **Application**. The primary key remains ApplicationID; **Checklist** was a weak entity and never had a primary key. The **Checklist** relation is therefore dropped and the Application relation becomes

Application (ApplicationID, ApplicationStatus, AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchAreas, HasPrimarySuper, PayMethConfirmed, EngProfConfirmed, RefereesConfirmed, LastToModifyStaffID, ProposedStartDate, ProposalSummary, RequireMoreInfo, DateAdded, DateLastChecked, DateLastModified, AwardType, PaymentMethod, flindersCampus, fullTime) **Primary key** ApplicationID

All other relations are as were in the previous section (1.4)

Removed the following possible relationships:

Relationship name	Identifying	Parent relation	Parent multiplicity	Foreign Keys	Child relation	Child multiplicity
Summarised by	Yes	Application	1	ApplicationID	Checklist	1

3.1.5.2 Mandatory on one side

Relationship name	Identifying	Parent relation	Parent multiplicity	Foreign Keys	Child relation	Child multiplicity
may have	No	Applicant	1	ApplicantID	VISA	0..1

The Visa application relation becomes

Visa (VisaID, ApplicantID, OriginCountryISOCode, VISAStatus, ValidFrom, ValidTo) **Primary key** VisaID **Foreign key** ApplicantID **references** Applicant(ApplicantID)

3.1.5.3 Optional on both sides

None in this case.

3.1.6 One-to-one recursive relationships

None in this case.

3.1.7 Superclass/subclass relationships

The following subclass/superclass relationship was removed at the conceptual phase due to the limitations of the modelling program not being able to handle such relationships.

University Staff member	-> RHD staff member
	-> Professional academic staff member (who can supervise)
	-> general academic staff member (cant supervise)

In this case the three child relations were merged into when with a 'can supervise' discriminator added. The distinction between a RHD staff member and a general staff member was found not to be needed and thusly dropped though if required a second discriminator such as 'is RHD manger' can be added . This resulted in the formation of the university staff member relation:

University Staff Member (StaffID, FName, LName, canSupervise)
Primary key StaffID

With all other relations remain as per the previous sections

3.1.8 Many-to-many binary relationship types

3.1.8.1 University Staff Member works in Research Area

Introduce the following relation:

University Staff Member_Research Area (StaffID, FORCode)
Primary key StaffID, FORCode
Foreign key StaffID **references** University Staff Member(StaffID)
Foreign key FORCode **references** Research Area(FORCode)

Introduce the following relationships:

Relationship name	Identifying	Parent relation	Parent multiplicity	Foreign Keys	Child relation	Child multiplicity
works in	Yes	University Staff Member	1	StaffID	University Staff Member_Research Area	0..*
works in2		Research Area	1	FORCode	University Staff Member_Research Area	0..*

This did not result in the modification of any other relations with all other relations remain as per the previous sections.

3.1.8.2 University Staff Member oversees Research Area

This relationship allows a staff member to manage applications in a particular research area for all those in that research area not just their own associated applications.

Introduce the following relation:

University Staff Member_Research Area2 (StaffID, FORCode)
Primary key StaffID, FORCode
Foreign key StaffID **references** University Staff Member(StaffID)
Foreign key FORCode **references** Research Area(FORCode)

Introduce the following relationships:

Relationship name	**Identifying**	**Parent relation**	**Parent multiplicity**	**Foreign Keys**	**Child relation**	**Child multiplicity**
Oversees	Yes	University Staff Member	1	StaffID	University Staff Member_Research Area2	0..*
Oversees2	Yes	Research Area	1	FORCode	University Staff Member_Research Area2	0..*

This did not result in the modification of any other relations with all other relations remain as per the previous sections.

3.1.8.3 Application in Research Area

Introduce the following relation:

Application_Research Area (ApplicationID, FORCode)
Primary key ApplicationID, FORCode
Foreign key ApplicationID **references** Application(ApplicationID)
Foreign key FORCode **references** Research Area(FORCode)

Introduce the following relationships:

Relationship name	**Identifying**	**Parent relation**	**Parent multiplicity**	**Foreign Keys**	**Child relation**	**Child multiplicity**
in		Application	1	ApplicationID	Application_Research Area	1..*
in2		Research Area	1	FORCode	Application_Research Area	0..*

This did not result in the modification of any other relations with all other relations remain as per the previous sections.

3.1.8.4 University Staff Member flags Application

Introduce the following relation:

University Staff Member_Application (StaffID, ApplicationID)
Primary key StaffID, ApplicationID
Foreign key StaffID **references** University Staff Member(StaffID)
Foreign key ApplicationID **references** Application(ApplicationID)

Introduce the following relationships:

Relationship name	Identifying	Parent relation	Parent multiplicity	Foreign Keys	Child relation	Child multiplicity
flags	Yes	University Staff Member	1	StaffID	University Staff Member_Application	0..*
flags2	Yes	Application	1	ApplicationID	University Staff Member_Application	0..*

This did not result in the modification of any other relations with all other relations remain as per the previous sections.

3.1.9 Complex relationship types

None in this case

3.1.10 Multi-valued attributes

None in this case

3.1.11 Document relations and foreign key attributes

Thus the full relations for this section in alphabetical order are

Applicant (ApplicantID, FName, LName, PrefTitle, Sex, DOB, StreetAddress, Suburb, Postcode, City, State, AddressCountryISOCode, Mobile, Phone, Email, NationalityCountryISOCode, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID, ManagedByStaffID)
Primary key ApplicantID
Alternate key (FName, LName, DOB, StreetAddress, Suburb, Postcode, City, State, Country)
Alternate key (FName, LName, DOB, Email)
Foreign key LastToModifyStaffID **references** University Staff Member(StaffID)

Application (ApplicationID, applicationStatus, AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchAreas, HasPrimarySuper, PayMethConfirmed, EngProfConfirmed, RefereesConfirmed, LastToModifyStaffID, ProposedStartDate, ProposalSummary, RequireMoreInfo, DateAdded, DateLastChecked, DateLastModified, awardType, flindersCampus, fullTime, paymentMethod) **Primary key** ApplicationID **Foreign key** ApplicantID **references** Application(ApplicationID) **Foreign key** LastToModifyStaffID **references** University Staff Member(StaffID) **Foreign key** ManagedByStaffID **references** University Staff Member(StaffID)
Application_Research Area (ApplicationID, FORCode) **Primary key** ApplicationID, FORCode **Foreign key** ApplicantID **references** Application(ApplicationID) **Foreign key** FORCode **references** Research Area(FORCode)
Correspondence (CorrID, Date, Summary, Message, ApplicantID, StaffID, CorrMeth, toStaff **Primary key** CorrID **Foreign key** ApplicantID **references** Applicant(ApplicantID) **Foreign key** StaffID **references** UniversityStaffMember(StaffID) **Foreign key** CorrMethod **references** Correspondence Method(Method)
Decision (DecID, Date, Comment, ApplicationID, StaffID, decType) **Primary key** DecID **Foreign key** ApplicationID **references** Application(ApplicationID) **Foreign key** StaffID **references** UniversityStaffMember(StaffID) **Foreign key** DecTypeID **references** Decision Type(type)
Decision Type (type) **Primary key** type
Degree (DegID, ApplicantID, Name, Type, YearCompleted, GPA, InstitutionName, InstitutionCountryISOCode) **Primary key** DegID **Foreign key** ApplicantID **references** Applicant(ApplicantID)
Document (DocID, UploadLink, Title, Description, DocType, DocStat, ApplicantID, ApplicationID) **Primary key** UploadLink **Foreign key** ApplicantID **references** Applicant(ApplicantID) **Foreign key** ApplicationID **references** Application(ApplicationID)
Publication (PubID, ApplicantID, Title, Publication, IssueNo, IssueDate, OnlineLink, OtherAuthorsNames, Language) **Primary key** PubID **Foreign key** ApplicantID **references** Applicant(ApplicantID)
Referee (RefID, ApplicationID, Name, Relation, Phone, Email, AcademicLink, EnglishSpeaker, EnglishLiterate)

Primary key RefID **Foreign key** ApplicationID **references** Application(ApplicationID)
Research Area (FORCode, Description, ResearchArea, GeneralArea) **Primary key** FORCode
Supervise as (StaffID, ApplicationID, PrimarySupervisor) **Primary key** StaffID, ApplicationID **Foreign key** ApplicationID **references** Application(ApplicationID) **Foreign key** StaffID **references** University Staff Member(StaffID)
University Staff Member (StaffID, FName, LName, canSupervise) **Primary key** StaffID
University Staff Member_Application (StaffID, ApplicationID) **Primary key** StaffID, ApplicationID **Foreign key** StaffID **references** University Staff Member(StaffID) **Foreign key** ApplicationID **references** Application(ApplicationID)
University Staff Member_Research Area (StaffID, FORCode) **Primary key** StaffID, FORCode **Foreign key** StaffID **references** University Staff Member (StaffID) **Foreign key** FORCode **references** Research Area(FORCode)
University Staff Member_Research Area2 (StaffID, FORCode) **Primary key** StaffID, FORCode **Foreign key** StaffID **references** University Staff Member (StaffID) **Foreign key** FORCode **references** Research Area(FORCode)
Visa (VisaID, ApplicantID, OriginCountryISOCode, VISAStatus, ValidFrom, ValidTo) **Primary key** VisaID **Foreign key** ApplicantID **references** Applicant(ApplicantID)

3.2 Normalisation

For the following tables IDs have been added for consistency with subsequent documents, in some cases this introduced full dependencies if there is a already a candidate key , however these have been ignored when defining normalisation.

Relation	**Functional Dependencies**	**Remarks**
Applicant	ApplicantID → FName, LName, PrefTitle, Sex, DOB, StreetAddress, Suburb, Postcode, City, State, Country, Mobile, Phone, Email, Nationality, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID FName, LName, DOB, StreetAddress, Suburb, Postcode, City, State, Country → ApplicantID, PrefTitle, Mobile, Phone, Email, Nationality, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID FName, LName, DOB, Email → ApplicantID, PrefTitle, StreetAddress, Suburb, Postcode, City, State, Country Mobile, Phone, Nationality, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes $\Rightarrow$ in 3NF
Application	ApplicationID → ApplicantID, AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchAreas, HasPrimarySuper, PayMethConfirmed, EngProfConfirmed, RefereesConfirmed, LastToModifyStaffID, ProposalDocID, AwardType, ManagedByStaffID, ProposedStartDate, ProposedSummary, RequireMoreInfo, DateAdded, DateLastChecked, DateLastModified, PayDocID, PaymentMethod, ApplicationStatus, flindersCampus, fullTime	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes $\Rightarrow$ in 3NF
Correspondence	CorrID → Date, Summary, Message, ApplicationID, StaffID, CorrMethID, toStaff	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes $\Rightarrow$ in 3NF
Decision	DecID → Date, Comment, ApplicationID, StaffID	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes $\Rightarrow$ in 3NF

Degree	DegID →ApplicantID, Name, Type, YearCompleted, GPA, InstitutionName, InstituitonCountry ApplicantID, Name → DegID, Type, YearCompleted, GPA, InstitutionName, InstituitonCountry	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes ⇒ in 3NF
Document*	DocID → UploadLink, DocStatus, DocumentType, Title, Description, ApplicantID, ApplicationID UploadLink → DocID, DocStatus, DocumentType, Title, Description, ApplicantID, ApplicationID	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes ⇒ in 3NF
Publication	PubID → ApplicantID, DocID, Title, Publication, IssueNo, IssueDate, OnlineLink, OtherAuthorsNames, Language	Attributes Publication, IssueNo, IssueDate, OnlineLink and OtherAuthorsNames are optional, therefore should not be considered as candidate keys. No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes ⇒ in 3NF
Referee	RefID → ApplicationID, Name, Relation, Phone, Email, Profession, AcademicLink, EnglishSpeaker, EnglishLiterate	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes ⇒ in 3NF
ResearchArea	FORCode → Description, ResearchArea, GeneralArea*	No repeating groups. has partial dependencies. (but is ignored due to increased complexity, otherwise would introduce a new GeneralArea relation) No transitive dependencies. Primary key fully determines non-primary-key attributes ⇒ in 2NF
University Staff Member	StaffID → LName, FName, canSupervise	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes ⇒ in 3NF

Visa	VisaID → ApplicantID, VisaStatus, OriginCountry, ValidFrom, ValidTo	No repeating groups. No partial dependencies. No transitive dependencies. Primary key fully determines non-primary-key attributes ⇒ in 3NF

The Relations introduced in section 1.8 to remove many-many relationships are all in 3NF or greater from as they include only two foreign keys which are not repeating groups, partially dependant on each other transiently dependant on each other.

3.2.1 Lookup relations[1]

In this schema there are several attributes with domains that allow only a small set of possible values. These attributes are Applicant(Nationality), Applicant(Country), Application(ApplicationStatus), Application(PaymentMethod), Document(DocumentStatus), Document(DocumentType), Visa(VisaStatus), Decision(DecisionType), Correspondence(CorrespondencMethod), and Award(AwardType).

In order to reduce the chance of update anomalies from keying-in incorrect values, we choose to create a lookup relation for each of these. In all cases, these relations are in one-to-many relationships with the relations containing their corresponding attributes. So, in each case, the lookup relation is designated the parent relation, the other relation is designated the child, and the primary key of the lookup table is posted to the child relation in place of the attribute.

This approach also has the benefit that users can: alter the associated description/types easily; or extend these domains to cover new cases as they arise.

ApplicationStatus (Status, Description) **Primary key** Status
Decision Type (Type) **Primary key** Type
CorrespondenceMethod (method) **Primary key** Method
AwardType (Type, Description, Method) **Primary key** Type
Payment Method (Method) **Primary key** Method
DocumentStatus (Status, Description)

[1] applied methodology as described in 'Chapter 18 Methodology – Monitoring and Tuning the Operational System Step 7.2 Duplicating non-key attributes in one-to-many (1:*) relationships to reduce joins'

Primary key Status
DocumentType (Type, Description) **Primary key** Type
Visa Status (Status, Description) **Primary key** Status
Country (CountryISOCode, Name) **Primary key** CountryISOCode

ALL these lookup relations have no repeated groups, partial dependencies or repeated groups and are thus in at least 3NF. It is also recognised that their introduction will introduce more joins in standard use but it is expected that this will make the management of applications and applicants more efficient, also maintaining the data integrity.

These were the added into their corresponding child relations (highlighted in blue in section 2.2)

3.2.1.1 Lookup relationships[2]

Relationship name	Identifying	Parent relation	Parent multiplicity	Foreign Keys	Child relation	Child multiplicity
lives in	No	Country	1..1	CountryISOCode→ AddressCountryISOCode	Applicant	0..*
nationality of	No	Country	0..1	CountryISOCode→ NationalityCountryISOCode	Applicant	0..*
has a	No	Visa Status	1..1	VisaStatus	Visa	0..*
originates in	No	Country	1..1	CountryISOCode → OriginCountryISOCode	Visa	0..*
has a	No	Document Status	1..1	Status → DocStat	Document	0..*
of	No	Document Type	1..1	Type → DocType	Document	0..*
will pay using	No	Payment Method	0..1	Method →paymentMethod	Application	0..*
has a	No	Application Status	1..1	Status → applicationStatus	Application	0..*
has a	No	Decision Type	1..1	type → decType	Decision	0..*

[2] applied methodology as described in 'Chapter 18 Methodology – Monitoring and Tuning the Operational System Step 7.2 Duplicating non-key attributes in one-to-many (1:*) relationships to reduce joins'

3.2.2 Normalised Relations

Applicant (ApplicantID, FName, LName, PrefTitle, Sex, DOB, StreetAddress, Suburb, Postcode, City, State, AddressCountryISOCode, Mobile, Phone, Email, NationalityCountryISOCode, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID)
Primary key ApplicantID
Alternate key (FName, LName, DOB, StreetAddress, Suburb, Postcode, City, State, Country)
Alternate key (FName, LName, DOB, Email)
Foreign key LastToModifyStaffID **references** University Staff Member(StaffID)
Foreign key AddressCountryISOCode **references** Country(CountryISOCode)
Foreign key NationalityCountryISOCode **references** Country(CountryISOCode)

Application (ApplicationID, applicationStatus, AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchAreas, HasPrimarySuper, PayMethConfirmed, EngProfConfirmed, RefereesConfirmed, LastToModifyStaffID, ProposedStartDate, ProposalSummary, RequireMoreInfo, DateAdded, DateLastChecked, DateLastModified, AwardType, flindersCampus, FullTime, PaymentMethod, ManagedByStaffID)
Primary key ApplicationID
Foreign key ApplicantID **references** Application(ApplicationID)
Foreign key LastToModifyStaffID **references** University Staff Member(StaffID)
Foreign key ManagedByStaffID **references** University Staff Member(StaffID)
Foreign key awardType **references** AwardType(Type)
Foreign key applicationStatus **references** Application Status(Status)
Foreign key paymentMethod **references** Payment Method(Method)

Application_Research Area (ApplicationID, FORCode)
Primary key ApplicationID, FORCode
Foreign key ApplicantID **references** Application(ApplicationID)
Foreign key FORCode **references** Research Area(FORCode)

Application Status (Status, Description)
Primary key Status

AwardType (Type, Description, Method)
Primary key Type

Correspondence (CorrID, Date, Summary, Message, ApplicantID, StaffID, CorrMeth, toStaff
Primary key CorrID
Foreign key ApplicantID **references** Applicant(ApplicantID)
Foreign key StaffID **references** UniversityStaffMember(StaffID)
Foreign key CorrMethod **references** Correspondence Method(Method)

Correspondence Method (Method)
Primary key Method

Country (CountryISOCode, Name)
Primary key CountryISOCode

Decision (DecID, Date, Comment, ApplicationID, StaffID, decType)
Primary key DecID
Foreign key ApplicationID **references** Application(ApplicationID)
Foreign key StaffID **references** UniversityStaffMember(StaffID)
Foreign key DecTypeID **references** Decision Type(type)

Decision Type (type)
Primary key type

Degree (DegID, ApplicantID, Name, Type, YearCompleted, GPA, InstitutionName, InstitutionCountryISOCode)
Primary key DegID
Foreign key ApplicantID **references** Applicant(ApplicantID)
Foreign key InstituitonCountryISOCode **references** Country(CountryISOCode)

Document (DocID, UploadLink, Title, Description, DocType, DocStat, ApplicantID, ApplicationID)
Primary key UploadLink
Foreign key DocType **references** DocumentType(Type)

Foreign key DocStat **references** DocumentStatus(Status) **Foreign key** ApplicantID **references** Applicant(ApplicantID) **Foreign key** ApplicationID **references** Application(ApplicationID)
DocumentStatus (Status, Description) **Primary key** Status
DocumentType (Type, Description) **Primary key** Type
Payment Method (Method) **Primary key** Method
Publication (PubID, ApplicantID, Title, Publication, IssueNo, IssueDate, OnlineLink, OtherAuthorsNames, Language) **Primary key** PubID **Foreign key** ApplicantID **references** Applicant(ApplicantID)
Referee (RefID, ApplicationID, Name, Relation, Phone, Email, AcademicLink, EnglishSpeaker, EnglishLiterate) **Primary key** RefID **Foreign key** ApplicationID **references** Application(ApplicationID)
Research Area (FORCode, Description, ResearchArea, GeneralArea) **Primary key** FORCode
Supervise as (StaffID, ApplicationID, PrimarySupervisor) **Primary key** StaffID, ApplicationID **Foreign key** ApplicationID **references** Application(ApplicationID) **Foreign key** StaffID **references** University Staff Member(StaffID)
University Staff Member (StaffID, FName, LName, canSupervise) **Primary key** StaffID
University Staff Member_Application (StaffID, ApplicationID) **Primary key** StaffID, ApplicationID **Foreign key** StaffID **references** University Staff Member(StaffID) **Foreign key** ApplicationID **references** Application(ApplicationID)
University Staff Member_Research Area (StaffID, FORCode) **Primary key** StaffID, FORCode **Foreign key** StaffID **references** University Staff Member (StaffID) **Foreign key** FORCode **references** Research Area(FORCode)
University Staff Member_Research Area2 (StaffID, FORCode) **Primary key** StaffID, FORCode **Foreign key** StaffID **references** University Staff Member (StaffID) **Foreign key** FORCode **references** Research Area(FORCode)
Visa (VisaID, ApplicantID, OriginCountryISOCode, VisaStatus, ValidFrom, ValidTo) **Primary key** VisaID **Foreign key** ApplicantID **references** Applicant(ApplicantID) **Foreign key** VisaStatus **references** Visa Status(Status) **Foreign key** OriginCountryISOCode **references** Country(CountryISOCode)
Visa Status (Status, Description) **Primary key** Status
Applicant (ApplicantID, FName, LName, PrefTitle, Sex, DOB, StreetAddress, Suburb, Postcode, City, State, AddressCountryISOCode, Mobile, Phone, Email, NationalityCountryISOCode, isNZAUCitizen, EnglishProficient, StudentID, DateAdded, LastToModifyStaffID) **Primary key** ApplicantID **Alternate key** (FName, LName, DOB, StreetAddress, Suburb, Postcode, City, State, Country) **Alternate key** (FName, LName, DOB, Email) **Foreign key** LastToModifyStaffID **references** University Staff Member(StaffID) **Foreign key** AddressCountryISOCode **references** Country(CountryISOCode) **Foreign key** NationalityCountryISOCode **references** Country(CountryISOCode)
Application (ApplicationID, applicationStatus, AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchAreas, HasPrimarySuper, PayMethConfirmed, EngProfConfirmed, RefereesConfirmed, LastToModifyStaffID, ProposedStartDate, ProposalSummary, RequireMoreInfo, DateAdded, DateLastChecked, DateLastModified, awardType, flindersCampus, fullTime, paymentMethod)

Primary key ApplicationID **Foreign key** ApplicantID **references** Application(ApplicationID) **Foreign key** LastToModifyStaffID **references** University Staff Member(StaffID) **Foreign key** ManagedByStaffID **references** University Staff Member(StaffID) **Foreign key** awardType **references** AwardType(Type) **Foreign key** applicationStatus **references** Application Status(Status) **Foreign key** paymentMethod **references** Payment Method(Method)
Application_Research Area (ApplicationID, FORCode) **Primary key** ApplicationID, FORCode **Foreign key** ApplicantID **references** Application(ApplicationID) **Foreign key** FORCode **references** Research Area(FORCode)
Application Status (Status, Description) **Primary key** Status
AwardType (Type, Description, Method) **Primary key** Type
Correspondence (CorrID, Date, Summary, Message, ApplicantID, StaffID, CorrMeth, toStaff **Primary key** CorrID **Foreign key** ApplicantID **references** Applicant(ApplicantID) **Foreign key** StaffID **references** UniversityStaffMember(StaffID) **Foreign key** CorrMethod **references** Correspondence Method(Method)
Correspondence Method (Method) **Primary key** Method
Country (CountryISOCode, Name) **Primary key** CountryISOCode
Decision (DecID, Date, Comment, ApplicationID, StaffID, decType) **Primary key** DecID **Foreign key** ApplicationID **references** Application(ApplicationID) **Foreign key** StaffID **references** UniversityStaffMember(StaffID) **Foreign key** DecTypeID **references** Decision Type(type)
Decision Type (type) **Primary key** type
Degree (DegID, ApplicantID, Name, Type, YearCompleted, GPA, InstitutionName, InstitutionCountryISOCode) **Primary key** DegID **Foreign key** ApplicantID **references** Applicant(ApplicantID) **Foreign key** InstituitonCountryISOCode **references** Country(CountryISOCode)
Document (DocID, UploadLink, Title, Description, DocType, DocStat, ApplicantID, ApplicationID) **Primary key** UploadLink **Foreign key** DocType **references** DocumentType(Type) **Foreign key** DocStat **references** DocumentStatus(Status) **Foreign key** ApplicantID **references** Applicant(ApplicantID) **Foreign key** ApplicationID **references** Application(ApplicationID)
DocumentStatus (Status, Description) **Primary key** Status
DocumentType (Type, Description) **Primary key** Type
Payment Method (Method) **Primary key** Method
Publication (PubID, ApplicantID, Title, Publication, IssueNo, IssueDate, OnlineLink, OtherAuthorsNames, Language) **Primary key** PubID **Foreign key** ApplicantID **references** Applicant(ApplicantID)
Referee (RefID, ApplicationID, Name, Relation, Phone, Email, AcademicLink, EnglishSpeaker, EnglishLiterate) **Primary key** RefID **Foreign key** ApplicationID **references** Application(ApplicationID)

Research Area (FORCode, Description, ResearchArea, GeneralArea)
Primary key FORCode

Supervise as (StaffID, ApplicationID, PrimarySupervisor)
Primary key StaffID, ApplicationID
Foreign key ApplicationID **references** Application(ApplicationID)
Foreign key StaffID **references** University Staff Member(StaffID)

University Staff Member (StaffID, FName, LName, canSupervise)
Primary key StaffID

University Staff Member_Application (StaffID, ApplicationID)
Primary key StaffID, ApplicationID
Foreign key StaffID **references** University Staff Member(StaffID)
Foreign key ApplicationID **references** Application(ApplicationID)

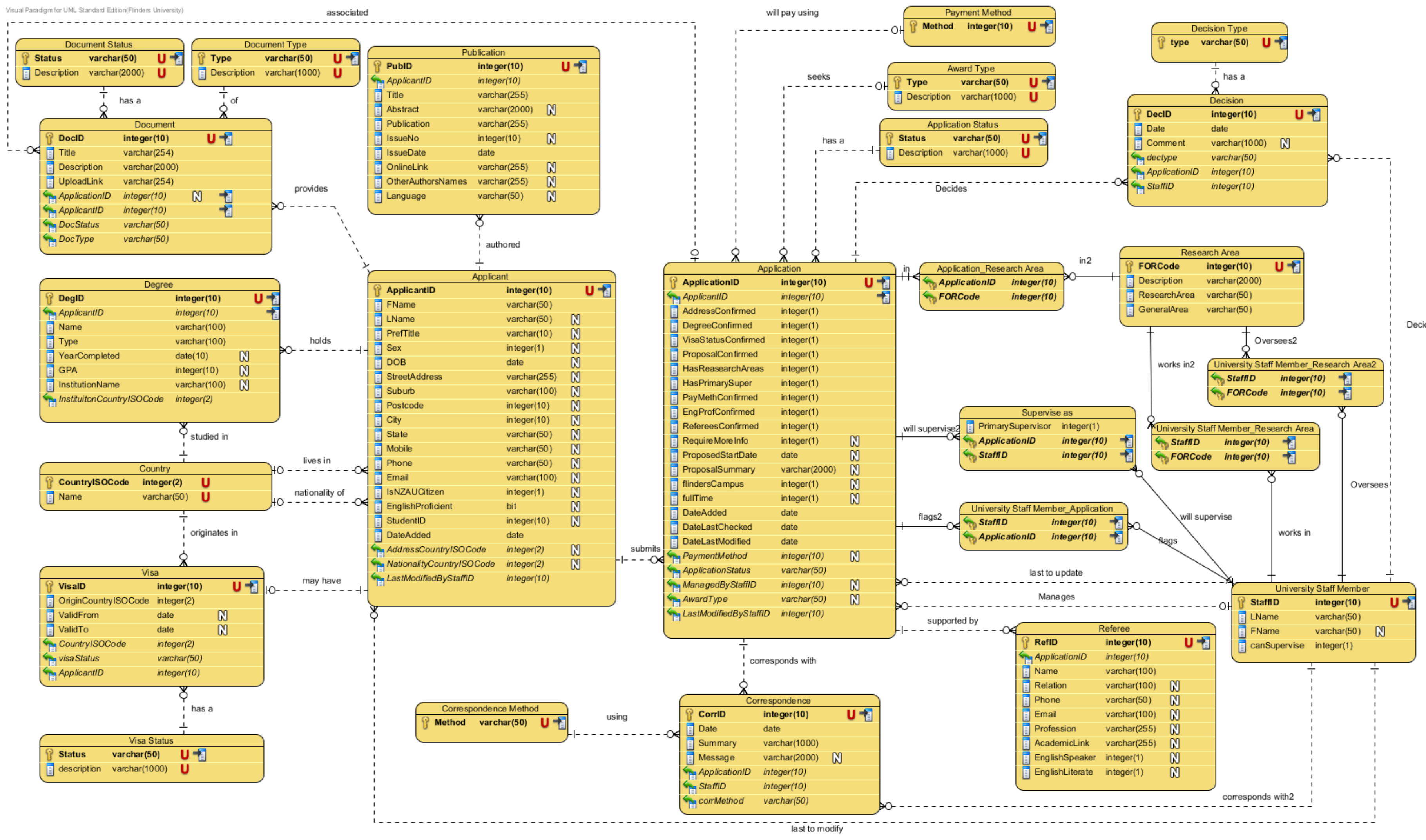
University Staff Member_Research Area (StaffID, FORCode)
Primary key StaffID, FORCode
Foreign key StaffID **references** University Staff Member (StaffID)
Foreign key FORCode **references** Research Area(FORCode)

University Staff Member_Research Area2 (StaffID, FORCode)
Primary key StaffID, FORCode
Foreign key StaffID **references** University Staff Member (StaffID)
Foreign key FORCode **references** Research Area(FORCode)

Visa (VisaID, ApplicantID, OriginCountryISOCode, VisaStatus, ValidFrom, ValidTo)
Primary key VisaID
Foreign key ApplicantID **references** Applicant(ApplicantID)
Foreign key VisaStatus **references** Visa Status(Status)
Foreign key OriginCountryISOCode **references** Country(CountryISOCode)

Visa Status (Status, Description)
Primary key Status

3.2.3 Logical E-R Diagram


Visual Paradigm for UML Standard Edition(Flinders University)
associated
will pay using
seeks
has a
Decides
provides
authored
holds
studied in
lives in
nationality of
originates in
may have
has a
submits
in
in2
Oversees2
works in2
will supervise2
flags2
will supervise
flags
works in
Oversees
Decides2
last to update
Manages
supported by
corresponds with
using
corresponds with2
last to modify
of
Document Status: Status varchar(50); Description varchar(2000)
Document Type: Type varchar(50); Description varchar(1000)
Publication: PubID integer(10); ApplicantID integer(10); Title varchar(255); Abstract varchar(2000); Publication varchar(255); IssueNo integer(10); IssueDate date; OnlineLink varchar(255); OtherAuthorsNames varchar(255); Language varchar(50)
Payment Method: Method integer(10)
Award Type: Type varchar(50); Description varchar(1000)
Application Status: Status varchar(50); Description varchar(1000)
Decision Type: type varchar(50)
Decision: DecID integer(10); Date date; Comment varchar(1000); dectype varchar(50); ApplicationID integer(10); StaffID integer(10)
Document: DocID integer(10); Title varchar(254); Description varchar(2000); UploadLink varchar(254); ApplicationID integer(10); ApplicantID integer(10); DocStatus varchar(50); DocType varchar(50)
Degree: DegID integer(10); ApplicantID integer(10); Name varchar(100); Type varchar(100); YearCompleted date(10); GPA integer(10); InstitutionName varchar(100); InstituitonCountryISOCode integer(2)
Country: CountryISOCode integer(2); Name varchar(50)
Visa: VisaID integer(10); OriginCountryISOCode integer(2); ValidFrom date; ValidTo date; CountryISOCode integer(2); visa Status varchar(50); ApplicantID integer(10)
Visa Status: Status varchar(50); description varchar(1000)
Applicant: ApplicantID integer(10); FName varchar(50); LName varchar(50); PrefTitle varchar(10); Sex integer(1); DOB date; StreetAddress varchar(255); Suburb varchar(100); Postcode integer(10); City integer(10); State varchar(50); Mobile varchar(50); Phone varchar(50); Email varchar(100); IsNZAUCitizen integer(1); EnglishProficient bit; StudentID integer(10); DateAdded date; AddressCountryISOCode integer(2); NationalityCountryISOCode integer(2); LastModifiedByStaffID integer(10)
Application: ApplicationID integer(10); ApplicantID integer(10); AddressConfirmed integer(1); DegreeConfirmed integer(1); VisaStatusConfirmed integer(1); ProposalConfirmed integer(1); HasReasearchAreas integer(1); HasPrimarySuper integer(1); PayMethConfirmed integer(1); EngProfConfirmed integer(1); RefereesConfirmed integer(1); RequireMoreInfo integer(1); ProposedStartDate date; ProposedSummary varchar(2000); flindersCampus integer(1); fullTime integer(1); DateAdded date; DateLastChecked date; DateLastModified date; PaymentMethod integer(10); ApplicationStatus varchar(50); ManagedByStaffID integer(10); AwardType varchar(50); LastModifiedByStaffID integer(10)
Application_Research Area: ApplicationID integer(10); FORCode integer(10)
Research Area: FORCode integer(10); Description varchar(2000); ResearchArea varchar(50); GeneralArea varchar(50)
University Staff Member_Research Area2: StaffID integer(10); FORCode integer(10)
Supervise as: PrimarySupervisor integer(1); ApplicationID integer(10); StaffID integer(10)
University Staff Member_Research Area: StaffID integer(10); FORCode integer(10)
University Staff Member_Application: StaffID integer(10); ApplicationID integer(10)
University Staff Member: StaffID integer(10); LName varchar(50); FName varchar(50); canSupervise integer(1)
Referee: RefID integer(10); ApplicationID integer(10); Name varchar(100); Relation varchar(100); Phone varchar(50); Email varchar(100); Profession varchar(255); AcademicLink varchar(255); EnglishSpeaker integer(1); EnglishLiterate integer(1)
Correspondence Method: Method varchar(50)
Correspondence: CorrID integer(10); Date date; Summary varchar(1000); Message varchar(2000); ApplicationID integer(10); StaffID integer(10); corrMethod varchar(50)

3.3 User transaction validation

University Staff Members

#	pathway
a)	Look up applicant + publications + degrees + visa Status + Associated documents by applicant name
b)	Look up applicant’s applications by applicant name
c)	Look up applicant’s applications by applicant email
d)	Look up incomplete applications
e)	Look up all correspondences relevant to an application
f)	Create new applicant and associated application records
g)	Look up which staff member updated an Application most recently
h)	Check for any decision recorded about an application
i)	Look up an existing application and attach a new standard type document to an application
j)	Look up an existing application and attached a new exceptional type document to an application
k)	Look up an existing application and list outstanding information (checklist).
l)	Update the checklist to confirm that a mandatory information requirement has been met
m)	Retrieve all on-going applications for which the user has made the most recent correspondence
n)	Record making a decision about an application
o)	Update the status of an application

Academic Staff

#	pathway
p)	Look up, add to, and delete from own current research areas
q)	Search for all applications in certain research areas that have been added since a certain time
r)	Flag interest in an application

RHD Co-ordination Staff

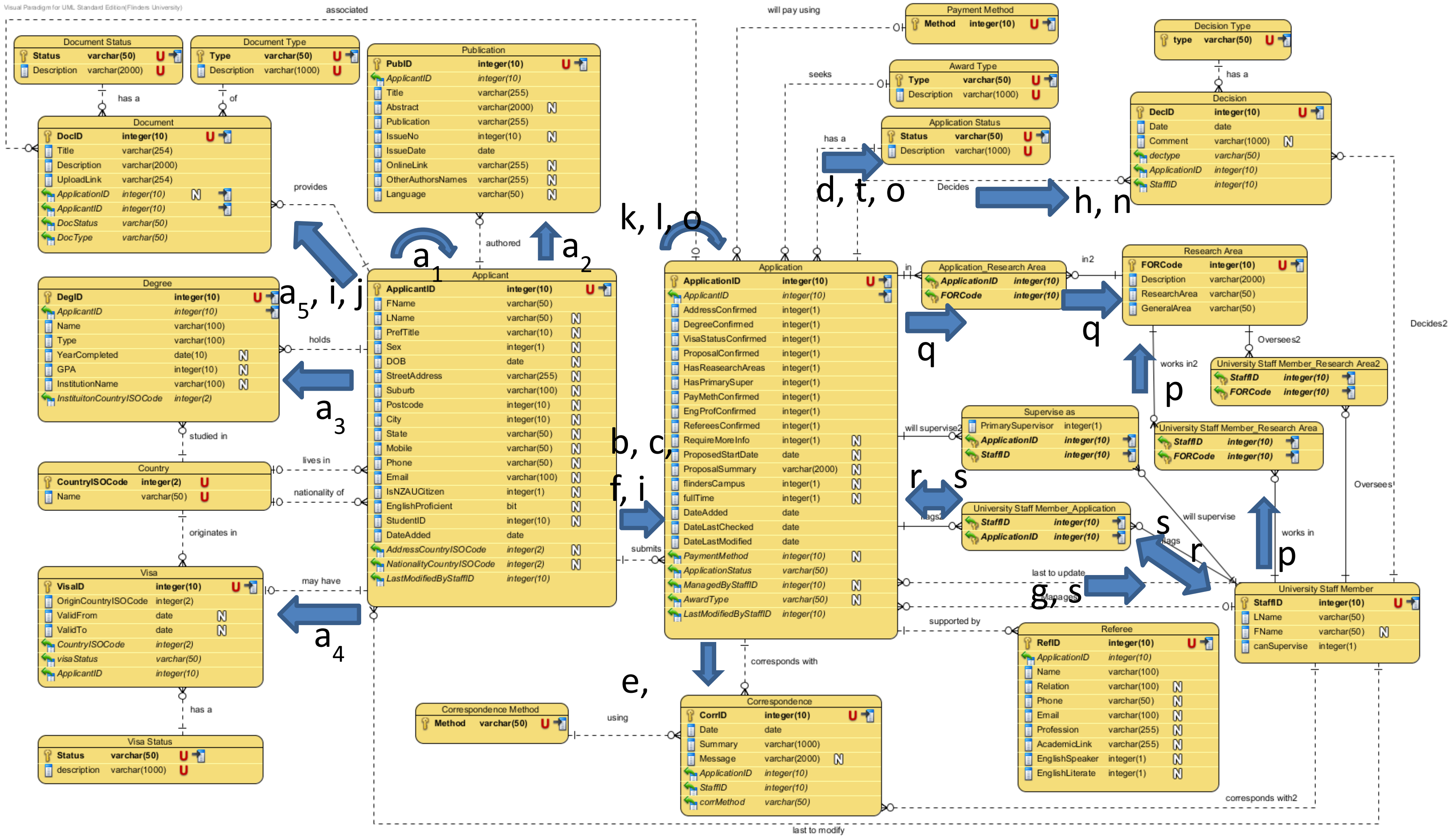
#	pathway
s)	Retrieve all staff who have flagged an application, or have edited an application or applicant record most recently
t)	Retrieve all ongoing applications

Note that the following modelling changes were triggered when validating user transactions:

- (e) – we had Correspondence in relationship with Applicant, but that could not record which correspondence was in regard to which application. We changed this to have Correspondence in relationship Application.

It should also be noted that the lookup relations will help facilitate in user transactions as entered data will be in a more consistent state i.e. document types will have set names rather than having to enter a new name in each time braking the integrity of the database.

3.3.1 Transaction pathways



3.4 Check integrity constraints

The following type of integrity constraints have been added to the logical model to protect the database from becoming incomplete, inaccurate, or inconsistent.

3.4.1 Required data

Required data has been specified through the use of not-null attributes identified in section 3 of the conceptual Documentation. These have been reviewed and included in the logical diagram. Since attributes in VP-UML are null-able by default, the following attributes were changed to not null, that is are required:

Applicant		**Degree**		**ResearchArea**	
fName	1.1	name	4.1	FORCode	9.1
sex	1.4	type	4.2	Description	9.2
email	1.15	**Document**		researchArea	9.3
dateAdded	1.20	Title	5.1	generalArea	9.4
Application		uploadLink	5.3	**University Staff Member**	
dateAdded	2.3	documentType	5.4	staffID	12.1
dateLastChecked	2.4	documentStatus	5.5	lName	12.3
dateLastModified	2.5	**Publication**		**Visa**	
awardType	2.7	title	7.1	VisaStatus	13.3
Checklist		Publication	7.3	**Correspondence**	
applicationStatus	3.1	issueDate	7.5	date	14.1
addressConfirmed	3.2	**Referee**		Summary/message	14.2
degreeConfirmed	3.3	name	8.1	corrMeth	14.4
visaConfirmed	3.4	relation	8.2	toStaff	14.5
proposalConfirmed	3.5	email	8.4	**Decision**	
engProfConfirmed	3.6	Profession	8.5	date	15.1
hasResearchAreas	3.7			decType	15.2
hasPrimarySuper	3.8			**SuperviseAs**	
payMethConfirmed	3.9			primarySupervisor	16.1
refereesConfirmed	3.10				

All other attributes not mentioned here are required to maintain integrity.

Note that some of these attributes are moved to separate entities in the next section, in these cases the foreign key will be not null. Furthermore any attributes added to one of these entities will also be not null and unique.

3.4.2 Attribute domain constraints

The attribute domains identified in section 4 of the conceptual documentation, have been introduced in the form of lookup relations, as produced in sections 2.2 and 2.3, and have been added to the logical diagram.

3.4.3 Multiplicity

The multiplicities identified in section 2 of the conceptual documentation and above in sections 1.3-17, have been added to the diagram. The additional multiplicities for the lookup tables identified in sections 2.2 and 2.3 have also been added to the logical diagram.

3.4.4 Entity integrity

The primary keys identified in section 5 of the conceptual documentation, have been reviewed in section 1 and added to/updated in the diagram. The additional primary keys for the attribute domain entities identified in section 4.2 have also been identified in section 1 and added to the diagram.

3.4.5 Referential integrity

The primary key of each parent relation has been added to the child relation in the form of a foreign key as specified by the Strong and weak entity table in sections 1.1, 1.2 and 1.7.

3.4.5.1 Null attributes (~insert rules)

It should be noted however that the attribute domain relations Identified in section 4.1 that replace null-able attributes will add their primary key as a null-able foreign keys in the child relation. More specifically Award Type, Country, Payment Method and Study Load and Location can all be null in the applicant/applicant entities because they may not be known at the time of the initial input of the entry.

3.4.5.2 Updates & Deletes

It is expected that there will be a significant amount of updates as an application (and applicant) progresses through the process. As such all entities have been set to cascade upon any update of the parent table. Even though it is expected that deletions will be rare (if at all) functionality has been included to maintain the integrity of the database. Deletions are performed such that

- All weak entities, that are children of applicant and or application only, CASCADE on delete excluding document which requires applicant to be deleted and
- All lookup tables and University Staff member relationships RESTRICT on all deletes.

This will enable all application/applicant specific material to be removed upon the deletion of an applicant and all application material removed upon a delete. Correspondingly the removal of statuses and other lookup tables (or staff tables) will not leave any of its children such as document without a status. This is particularly relevant for application status as the deletion of a status, will make it lose its point in the process.

RESTRICT has been chosen as it reflects the requirement that nothing should be deleted, since it is possible that mistakes and repeated data can be entered some deletion is enable in this way.

3.4.6 General constraints.

Currently there are no high level constraints that will affect the validity of the logical model. If any are later realised that will be discovered during Review or the final checks of the physical diagram.

There are however pseudo constraints through the checklist (now merged into the application) whereby an application cannot have certain application statuses unless the corresponding checklist attributes are true.

3.4.7 Document all integrity constraints

All integrity constraints have been applied to the logical model, which can be used to produce a data dictionary when required.

3.5 Review logical data model

The Logical data model will be reviewed with Paul, the head of the Research Higher Degree Office to ensure that it meets all their requirements and is a true (or as close to possible) representation of the data requirements as specified by the Research Higher Degree Office and the staff members who will use the database.

Some of the areas of specific mention that we have included in the data base are

- The use of lookup tables for better data consistency and integrity
- A flat more variable document relation enabling documents of specific types to be added and not necessarily linked to a specific relation of an applicant.
- Nearly all of the database relations are in 3NF with some divergences such as the above added to increase efficiency

3.6 Check for future growth

It has been found that there are several possible areas that the logical database model may need to include in future. These include but are not limited to

- Expanding research areas, within the school of CSEM or the wider university,
- The possible inclusion of a housing & transport weak entity/attribute for the application/applicant,
- The inclusion of some form of disability/issues entity and or
- The addition of other statuses, types or other lookup tuples.

It has been found that these areas can be accounted for, in the form of adding new entities or attributes. However some possible future growths areas such as the reuse of referees for anther applications (by the same applicant) will break the model and force a redesign. Even though the redesign will be minor, creating a RefereeApplicaiton entity of referee and application IDs and replacing applicationID with applicantID (if maintaining connection with the applicant), it will mean the database and any programs will have to go offline while the modifications are made.

It has been assumed that such cases occur rarely and as such re-entering details will not be too inefficient. This will ensure that any detail changes that occur between applications are added as different entries.

3.7 Develop Test Plan

3.7.1 Introduction

General description: The database will from the back bone of the application storage for prospective Research Higher Degree (RHD) students. The database will allow academic staff, research higher degree staff and database administrator's access to create, update and mange RHD applications.

Mission statement: To produce a database that can be used to manage the initial applications of prospective Research Higher Degree Students

Schedule
To write the database Definition
To find data and write scripts to populate the database
To write scripts that mimic possible transactions
To then run the scripts automatically to ensure transactions are possible and operate as desired
Iteratively add more data, possible transaction scripts and continue testing until all of the core transactions are covered and the majority of the infrequent transactions have some level of minimum coverage

3.7.2 Required resources

Type	**What**	**Why**
Software	MySQL Community Edition	The DBMS platform the DB will run on
	Text Editor	To write and edit scripts
Hardware	Two computers	To run MySQL and the text editor
Testing Tools	STKUnit	To automate testing of the scripts
Staff	2 people	To both write and check the scripts

3.7.3 Testable aspects

Unit tested transactions

- All the user transactions, pathways mentioned above in section 3.3 on page 57.

3.7.4 Non Testable aspects

- Any graphical user interface that uses these the database will not be tested
- Highly specific performance enhancements
- Maintenance transactions (Deletions, though the required functionality has been implemented)

3.7.5 Test Produced documents

A single test document, listing

- The specifics of each test, what they are trying to test
- Assumptions
- Demonstration that the test pass

3.7.6 Risks and dependencies

- It is not possible to test the database as it will be in a few yeas time and the issues it may have, due to the impracticality of populating the database with such a large volume of information.
- It is not possible to test all transaction pathways as many are complicated and will be used rarely if at all

3.7.7 Project Criteria

Goal: To ensure that the database can handle the main transactions as outlined in previous document in line with the overall requirements of the database.

3.7.8 Success and Failure measured by

Success will be measured based on the amount of tests that pass combined with their relevance to the core transactions of the database

3.8 CHANGE LOG

a) Added intermediate relationships, corrected some difference between how terms are stated here and subsequent documents increasing consistency.
b) Added notes about normalising many-many introduced relations and lookup relations

4 Physical Design and SQL scripts

4.1 Translate logical data model for target DBMS

4.1.1 Select target DBMS

The target DBMS has been mandated by the customers as MySQL v5.1. Since this was known to be the target DBMS at the initialisation of the project, the previous two models that of the conceptual and logical have been designed to be compatible with MySQL, with limited amount of specific implementation required. This is seen (though noted not technically correct) in the form of adding variables applicable to MySQL (such as setting variables as VarChars in place of more general Strings variables) in the conceptual diagram.

The information gathered the in the previous three sections of requirements gathering and analysis, Conceptual model Diagram and documentation and logical model Diagram and documentation in their latest iteration have been reviewed and collected into a single information source.

The target, MySQL, DBMS has been studied revealing how to preform base transactions (such that Create, Read Update and Delete Base Relations are done for the most part through standard SQL (Oracle Inc. 2014a) and that most, if not all of the required functionality (that of. Keys, Domains and constraints) is available through the standard enterprise version as will be used in the final implementation of the database.

This was then used to produce the following **Relational database schema**

SCHEMA HERE MAYBE?

4.1.2 Design base relations

Implementing base relations

The data base relation have been implemented using ISO SQL standard (Section 6.1) with some specific minor MySQL specific adjustments. Note that while MySQL accepts sstandard SQL syntax, it does ignore certain constructs, thus doe not provide a full set of SQL features. For example, the CHECK constraint syntax, normally used to implement genral constraints, is accepted but ignored (Oracle Inc. 2014b).

In implementating the base relations the following was adhered to

Document design of base relations

Table 4 Base relations definitions in DataBase Definition Language

Domain	AuthorList	variable length character string, max length 255
Domain	Language	variable length character string, max length 50
Domain	DegreeName	variable length character string, max length 100
Domain	DegreeType	variable length character string, max length 100
Domain	GPA	unsigned integer, in the range 0-7
Domain	InstitutionName	variable length character string, max length 100
Domain	CountryISOCode	fixed length character string, of width 2
Domain	PersonNameWhole	variable length character string, max length 100
Domain	RelationName	variable length character string, max length 100
Domain	PhoneNumber	variable length character string, max length 50
Domain	Profession	variable length character string, max length 255

Domain	MediaFilePath	variable length character string, max length 255
Domain	PersonNamePart	variable length character string, max length 50
Domain	PersonTitle	variable length character string, max length 10
Domain	Binary	unsigned integer, in the range 0-1
Domain	StreetAddress	variable length character string, max length 255
Domain	SuburbName	variable length character string, max length 100
Domain	Postcode	unsigned integer, in the range 0-10,000,000
Domain	CityName	variable length character string, max length 50
Domain	StateName	variable length character string, max length 50
Domain	ResearchArea	variable length character string, max length 50
Domain	FORCode	unsigned integer, in the range 0-999,999

```
Decision Type (
  DecisionTypeID IdInteger NOT NULL AUTO_INCREMENT,
  type TypeLabelShort NOT NULL UNIQUE
                comment 'the type of decision/comment made',
  PRIMARY KEY (DecisionTypeID)
) comment='the possible types of decisions/comments that can be made'


Payment Method (
  PayMethodID IdInteger NOT NULL AUTO_INCREMENT,
  Method TypeLabelShort NOT NULL UNIQUE
    comment 'Method of payment, e.g. scholarship, letter of financial support etc.'),
  PRIMARY KEY (PayMethodID)
)
comment='the possible payment methods of a Research higher degree'

Application (
  ApplicationID IdInteger NOT NULL AUTO_INCREMENT,
  ApplicantEmail Email
    comment 'Denormalised column to improve retrieval of applications via email address',
  ApplicantID IdInteger NOT NULL
    comment 'the ID of the applicant who proposed this application',
  AddressConfirmed tinyint(1) NOT NULL comment 'All contact details appear valid',
  DegreeConfirmed Boolean NOT NULL
    comment 'The degree is a recognised degree of the institution',
  VisaStatusConfirmed Boolean NOT NULL
    comment 'The visa status is backed by an official document',
  ProposalConfirmed Boolean NOT NULL
    comment 'The proposal is contains appropriate detail',
  HasResearchAreas Boolean NOT NULL
    comment 'Has nominated research areas relevant to the proposal',
  HasPrimarySuper Boolean NOT NULL
    comment 'Has the required number of supervisors',
  PayMethConfirmed Boolean NOT NULL
    comment 'The payment method is backed by an official document',
  EngProfConfirmed Boolean NOT NULL
    comment 'The applicant has some level of English literacy',
  RefereesConfirmed Boolean NOT NULL
    comment 'The referees details appear to be correct',
  RequireMoreInfo Boolean,
  ProposedStartDate Date
    comment 'The date the applicant prefers to start the RHD (Entered as 1/1/## for S1 and 1/7/## for S2)',
  ProposalSummary ProposalSummary comment 'What the proposal is about',
  flindersCampus Boolean
    comment 'the applicant wants to complete the degree a main campus',
  fullTime Boolean comment 'the applicant wants to undergo the degree full time',
  DateAdded Date NOT NULL comment 'the date the application was added',
  DateLastChecked Date NOT NULL
    comment 'the date the application was last checked',
  DateLastModified Date NOT NULL
    comment 'the date the application was last modified',
  ManagedByStaffID IdInteger
    comment 'The staff ID of the staff member who has been personally assigned to manage this application
(since it may not be used it is nullable)',
  LastModifiedByStaffID IdInteger NOT NULL
    comment 'the staff member ID of the last person to modify the application (all modifications are recorded in
the decision table)',
  applicationStatusID IdInteger NOT NULL,
  awardID IdInteger,
  PayMethodID IdInteger,
  PRIMARY KEY (ApplicationID),
  FOREIGN KEY (PayMethodID) REFERENCES `Payment Method` (PayMethodID)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (ApplicantID) REFERENCES Applicant (ApplicantID)
    ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (applicationStatusID)
    REFERENCES `Application Status` (ApplicationStatusID)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (ManagedByStaffID) REFERENCES `University Staff Member` (StaffID)
    ON UPDATE Cascade ON DELETE Set null,
  FOREIGN KEY (awardID) REFERENCES `Award Type` (AwardID)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (LastModifiedByStaffID)
    REFERENCES `University Staff Member` (StaffID)
    ON UPDATE Cascade ON DELETE Restrict
)
comment='Holds the application details including a checklist of recorded information and applicaiton stage'

Document Status (
  DocStatusID IdInteger NOT NULL AUTO_INCREMENT,
  Status TypeLabelShort NOT NULL UNIQUE
    comment 'Official and translation status of a document associated to an Applicant',
  Description DescLong NOT NULL
    comment 'the details and implication of this status',
  PRIMARY KEY (DocStatusID)
)
comment='the possible statuses of a document'

Publication (
  PubID IdInteger NOT NULL AUTO_INCREMENT
    comment 'The primary key that uniquely identifies the publication',
  ApplicantID IdInteger NOT NULL
    comment 'the ID of the applicant who authored the publication',
  Title DocumentTitle NOT NULL comment 'The title of the publication',
  Abstract DescLong comment 'A abstract/description of the publication',
  Publication DocumentTitle NOT NULL comment 'The journal/magazine publisher ',
  IssueNo IdInteger comment 'The issue/edition number of the publication',
  IssueDate Date NOT NULL comment 'The date the publication was issued',
  OnlineLink URL comment 'An online link to the publication',
  OtherAuthorsNames AuthorList comment 'Other authors of the publication',
  Language Language comment 'language of the publication',
  PRIMARY KEY (PubID),
  FOREIGN KEY (ApplicantID) REFERENCES Applicant (ApplicantID)
```

```
    ON UPDATE Cascade ON DELETE Cascade
)

Degree (
  DegID IdInteger NOT NULL AUTO_INCREMENT
    comment 'The primary key that uniquely identifies the degree',
  ApplicantID IdInteger NOT NULL
    comment 'the ID of the applicant who holds this degree',
  Name DegreeName NOT NULL comment 'The title of the degree',
  Type DegreeType NOT NULL
    comment 'The type of the degree-Could add specific types',
  YearCompleted Date
    comment 'The year the degree was completed or will be completed',
  GPA GPA comment 'The GPA of the degree',
  InstitutionName InstitutionName comment 'The name of the institution',
  InstitutionCountryISOCode CountryISOCode NOT NULL
    comment 'the country the institution is based in',
  PRIMARY KEY (DegID),
  FOREIGN KEY (InstitutionCountryISOCode) REFERENCES Country (CountryISOCode)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (ApplicantID) REFERENCES Applicant (ApplicantID)
    ON UPDATE Cascade ON DELETE Cascade
)
comment='Any Degrees already held by the applicant'

Referee (
  RefID IdInteger NOT NULL AUTO_INCREMENT
    comment 'The primary key that uniquely identifies the referee supporting an application',
  ApplicationID IdInteger NOT NULL,
  Name PersonNameWhole NOT NULL comment 'The full name of the referee',
  Relation RelationName comment 'The referees relation to the applicant',
  Phone PhoneNumber comment 'The referees phone number',
  Email Email comment 'The referees email address',
  Profession Profession comment 'The referees profession',
  AcademicLink URL comment 'The referees professional page (linked in or university)',
  EnglishSpeaker Boolean comment 'If the Referee can speak English',
  EnglishLiterate Boolean comment 'If the Referee can read and write in English',
  PRIMARY KEY (RefID),
  FOREIGN KEY (ApplicationID) REFERENCES Application (ApplicationID)
    ON UPDATE Cascade ON DELETE Cascade
)
comment='A referee for a application'

Document (
  DocID IdInteger NOT NULL AUTO_INCREMENT
    comment 'The primary key that uniquely identifies the document',
  Title DocumentTitle NOT NULL comment 'The title of the document',
  Description DescLong NOT NULL
    comment 'A specific summary related to the document i.e. valid till 2015 etc.',
  UploadLink MediaFilePath NOT NULL
    comment 'An link to a version uploaded and stored on the university servers',
  ApplicationID IdInteger,
  ApplicantID IdInteger NOT NULL comment 'the ID of the applicant who provided this document',
  DocStatusID IdInteger NOT NULL,
  DocTypeID IdInteger NOT NULL,
  PRIMARY KEY (DocID),
  FOREIGN KEY (ApplicationID)  REFERENCES Application (ApplicationID)
```

```
   ON UPDATE Cascade ON DELETE Restrict
  FOREIGN KEY (ApplicantID) REFERENCES Applicant (ApplicantID)
   ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (DocStatusID) REFERENCES `Document Status` (DocStatusID)
   ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (DocTypeID) REFERENCES `Document Type` (DocTypeID)
   ON UPDATE Cascade ON DELETE Restrict
)
comment='Links to any relevant documents along with descriptions,
  types and statuses.'

Applicant (
  ApplicantID IdInteger NOT NULL AUTO_INCREMENT
   comment 'The primary key that uniquely identifies the applicant',
  FName PersonNamePart NOT NULL comment 'First name',
  LName PersonNamePart comment 'Last name',
  PrefTitle PersonTitle comment 'Title Mr,  Mrs,  Miss,  Dr. *',
  Sex Binary comment 'The sex of the applicant',
  DOB Date comment 'Date of birth',
  StreetAddress StreetAddress
   comment 'Residence number and street of residence',
  Suburb SuburbName comment 'The suburb of residence',
  Postcode Postcode comment 'The postcode of residence',
  City CityName comment 'The city or town of residence',
  State StateName comment 'The State of residence',
  Mobile PhoneNumber comment 'Mobile phone number',
  Phone PhoneNumber comment 'Landline phone number',
  Email Email comment 'The email address of the applicant',
  IsNZAUCitizen Boolean
   comment 'Is a new Zealand or Australian citizen, a check to see if visa information is required ***',
  EnglishProficient Boolean comment 'English ability',
  StudentID IdInteger comment 'The flinders university student id if they are or have been enrolled at flinders
university',
  DateAdded Date NOT NULL comment 'The date the applicant was added to the system',
  AddressCountryISOCode CountryISOCode,
  NationalityCountryISOCode CountryISOCode,
  LastModifiedByStaffID IdInteger NOT NULL
   comment 'the staff member ID of the last person to modify the applicant (all modifications are recorded in
the decision table)',
  PRIMARY KEY (ApplicantID),
  FOREIGN KEY (AddressCountryISOCode) REFERENCES Country (CountryISOCode)
   ON UPDATE Cascade ON DELETE Restrict
  FOREIGN KEY (NationalityCountryISOCode)
   REFERENCES Country (CountryISOCode) ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (LastModifiedByStaffID)
   REFERENCES `University Staff Member` (StaffID)
   ON UPDATE Cascade ON DELETE Restrict
)
comment='Holds the applicant specific details'

Visa (
  VisaID IdInteger NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely identifies the visa',
  ValidFrom Date comment 'When the visa is valid from',
  ValidTo Date comment 'When the visa is valid to',
  CountryISOCode CountryISOCode NOT NULL comment 'the applicant country, the visa is granted to',
  ApplicantID IdInteger NOT NULL comment 'the ID of the applicant who holds or may hold this visa',
  VisaStatusID IdInteger NOT NULL,
  PRIMARY KEY (VisaID),
  FOREIGN KEY (CountryISOCode) REFERENCES Country (CountryISOCode)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (VisaStatusID) REFERENCES `Visa Status` (VisaStatusID)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (ApplicantID) REFERENCES Applicant (ApplicantID)
    ON UPDATE Cascade ON DELETE Cascade
)
comment='The applicants visa details'

Visa Status (
  VisaStatusID IdInteger NOT NULL AUTO_INCREMENT,
  Status TypeLabelShort NOT NULL UNIQUE comment 'the status of the visa application',
  description DescMedium NOT NULL comment 'a description of the status of the visa',
  PRIMARY KEY (VisaStatusID)
)
comment='the possible statuses of the visa application'

Correspondence (
  CorrID IdInteger NOT NULL AUTO_INCREMENT,
  `Date` Date NOT NULL comment 'The date the correspondence was made/received',
  Summary DescMedium NOT NULL comment 'A small summary of the Correspondence',
  Message DescLong comment 'The actual message contained in the correspondence',
  ApplicationID IdInteger NOT NULL comment 'the application ID the correspondence is in relation to',
  StaffID IdInteger NOT NULL comment 'the staff ID of the staff member the correspondence is to/from',
  CorrMethodID IdInteger NOT NULL,
  PRIMARY KEY (CorrID),
  FOREIGN KEY (ApplicationID) REFERENCES Application (ApplicationID)
    ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (CorrMethodID) REFERENCES `Correspondence Method` (CorrMethodID)
    ON UPDATE Cascade ON DELETE Restrict
)
comment='Correspondence between the Applicant and University Staff Member'

Decision (
  DecID IdInteger NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely identifies the
Decision/comment made',
  `Date` Date NOT NULL comment 'The date the decision was made on',
  Comment DescMedium comment 'Extra information about the decision',
  ApplicationID IdInteger comment 'the id of the application this decision is made with regards to, if this is a
decision associated with an application',
  StaffID IdInteger NOT NULL comment 'the staff ID of the staff member who made this decision/comment',
  DecisionTypeID IdInteger NOT NULL,
  Reportable Boolean NOT NULL comment 'a boolean that is automatically ticked if the change is deemed
reportable (status changes request filled etc.)',
  Sent Boolean comment 'a boolean to check if the related email has been sent',
  PRIMARY KEY (DecID),
  FOREIGN KEY (DecisionTypeID) REFERENCES `Decision Type` (DecisionTypeID)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (ApplicationID) REFERENCES Application (ApplicationID)
    ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (StaffID) REFERENCES `University Staff Member` (StaffID)
    ON UPDATE Cascade ON DELETE Restrict
)
comment='The decision/comment made for an application by a RHD staff member'

Research Area (
  FORCode IdInteger NOT NULL AUTO_INCREMENT comment 'The Australian Field Of Research (FOR) code,
primary key,  that uniquely identifies the Research area',
  Description DescLong NOT NULL comment 'A small text description of the FOR i.e. 2201',
  ResearchArea ResearchArea NOT NULL comment 'The FOR title; i.e. Applied Ethics',
  GeneralArea ResearchArea NOT NULL comment 'The general area of the FOR',
  PRIMARY KEY (FORCode)
)

University Staff Member (
  StaffID IdInteger NOT NULL AUTO_INCREMENT comment 'The flinders uni staff ID number,
  the primary key that uniquely identifies the staff member',
  FName PersonNamePart NOT NULL comment 'The last name of the staff member',
  LName PersonNamePart comment 'The first name of the staff member',
  canSupervise Boolean NOT NULL comment 'if the staff member is able supervise a RHD applicant',
  email Email NOT NULL,
  PRIMARY KEY (StaffID)
)
comment='a university staff member who may be able to supervise a application'

Correspondence Method (
  CorrMethodID IdInteger NOT NULL AUTO_INCREMENT,
  Method TypeLabelShort NOT NULL UNIQUE comment 'the method of correspondence',
  PRIMARY KEY (CorrMethodID)
)

Application Status (
  ApplicationStatusID IdInteger NOT NULL AUTO_INCREMENT,
  Status TypeLabelShort NOT NULL UNIQUE comment 'the name of the status',
  Description DescMedium NOT NULL comment 'a full description of the status',
  PRIMARY KEY (ApplicationStatusID),
)
comment='the possible application statuses of an application'

Document Type (
  DocTypeID IdInteger NOT NULL AUTO_INCREMENT,
  Type TypeLabelShort NOT NULL UNIQUE comment 'the type of document eg. Publication, visa etc.',
  Description DescMedium NOT NULL comment 'a full description of the type of the document',
  PRIMARY KEY (DocTypeID)
)

Award Type (
  AwardID IdInteger NOT NULL AUTO_INCREMENT,
  Type TypeLabelShort NOT NULL UNIQUE comment 'the type of award sought by the applicant',
  Description DescMedium NOT NULL comment 'a full description of the award type',
  PRIMARY KEY (AwardID)
)
comment='the possible award types (degrees) sought by an application'

Country (
  CountryISOCode CountryISOCode NOT NULL
    comment 'the country corresponding ISO 3166-1 alpha-2 code',
  Name TypeLabelShort NOT NULL UNIQUE comment 'the full name of the country',
  PRIMARY KEY (CountryISOCode)
)
comment='a list of countries for reuse in nationality,
  institution country,  address and visa country'
```

```
Supervise as (
  PrimarySupervisor Boolean NOT NULL comment 'if the supervisor is a primary',
  ApplicationID IdInteger NOT NULL comment 'the application ID of the application the staff member will
supervise',
  StaffID IdInteger NOT NULL comment 'the staff ID of the staff member who will supervise the applicaiton',
  PRIMARY KEY (ApplicationID,  StaffID),
  FOREIGN KEY (ApplicationID) REFERENCES Application (ApplicationID)
    ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (StaffID) REFERENCES `University Staff Member` (StaffID)
    ON UPDATE Cascade ON DELETE Cascade
)
comment='the staff members who have agreed to supervise an application'

Application_Research Area (
  ApplicationID IdInteger NOT NULL comment 'the ID of the application',
  FORCode FORCode NOT NULL comment 'the FORCode research area the applicant states they want to study
in',
  PRIMARY KEY (ApplicationID, FORCode),
  FOREIGN KEY (ApplicationID) REFERENCES Application (ApplicationID)
    ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (FORCode) REFERENCES `Research Area` (FORCode)
    ON UPDATE Cascade ON DELETE Restrict
)
comment='The application and the research area they are looking to study in'

University Staff Member_Research Area (
  StaffID IdInteger NOT NULL
    comment 'the staff ID of the staff member who works in the research area',
  FORCode FORCode NOT NULL
    comment 'the field of research that the staff member works in',
  PRIMARY KEY (StaffID, FORCode),
  FOREIGN KEY (StaffID) REFERENCES `University Staff Member` (StaffID)
    ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (FORCode) REFERENCES `Research Area` (FORCode)
    ON UPDATE Cascade ON DELETE Restrict
)
comment='the research area the staff member states they work in'

University Staff Member_Research Area2 (
  StaffID IdInteger NOT NULL
    comment 'the staff ID of the staff member who oversees the research area',
  FORCode FORCode NOT NULL
    comment 'the FORCode of the research area that the staff member oversees',
  PRIMARY KEY (StaffID, FORCode),
  FOREIGN KEY (StaffID) REFERENCES `University Staff Member` (StaffID)
    ON UPDATE Cascade ON DELETE Restrict,
  FOREIGN KEY (FORCode) REFERENCES `Research Area` (FORCode)
    ON UPDATE Cascade ON DELETE Cascade
)
comment='the research areas that a staff member oversees (can be more than one staff member per area)'

University Staff Member_Application (
  StaffID IdInteger NOT NULL comment 'the staff ID of the staff member who flagged the application',
  ApplicationID IdInteger NOT NULL comment 'the application the staff member has flagged',
  ReceiveEmailUpdates Boolean NOT NULL comment 'a boolean that a user can check if they want to be alerted
About an update or simply keep a reference on their application page',
  PRIMARY KEY (StaffID,  ApplicationID),
  FOREIGN KEY (StaffID) REFERENCES `University Staff Member` (StaffID)
    ON UPDATE Cascade ON DELETE Cascade,
  FOREIGN KEY (ApplicationID) REFERENCES Application (ApplicationID)
    ON UPDATE Cascade ON DELETE Cascade
)
comment='the staff members who have flagged an application to come back and check later'
```

4.1.3 Design representation of derived data

No derived data fields have been identified in this design.

There are some fields that we assume will be automatically populated by the controlling application, such as the most recent modification date of and application and the staff member responsible for a change.

There are other important derivable values, but these will not be stored by the system. This includes the age of an application, number of applications flagged, number of applications managed, and application history. To aid such queries, indexes have been placed on the relevant foreign and primary keys that are expected to be used often.

Note that the checklist is under manual control of users and not automatically derived.

4.1.4 Design general constraints

There are a few design constraints that have been implemented, these are

- There should be a maximum of one primary supervisor for each application.
- Only staff members that have attribute canSupervise marked as true should be able to participate in the ‘Supervise as’ relation.

As MySQL does not implement CHECK constraint freatures of standard SQL (Oracle Inc. 2014b), we use customised triggers to ensure that there is only one primary supervisor for each application. The implementation of these triggers is shown in section 4.6.4 on page 103. These implementation of triggers to enfore the constraint over participation in the ‘Supervise as’ relation has been left as future work, should violations of this constraint prove to be problematic in the running system.

General constraints such as text and dates being non empty is assumed to be enforced by the application interface. This is mainly because MySQL does not support constraints or partial indexes making enforcing of such constraints difficult. However there are workarounds using ‘Trigger’s but since these can have a significant impact on performance (running every time on an update/insert event if implemented in most cases) they have not be included unless deemed absolutely necessary.

4.2 Design file organisations and indices

4.2.1 Analyse transactions

In this section we analyse the most frequent transactions with respect to which base relations they operate on and which operations are performed. This will serve to guide us as to where we can expect bottlenecks in the running system, and therefore what will be the most appropriate design decisions to ensure good system performance.
The user transactions are shown in Table 5, Table 6 and Table 7. These transactions are cross-referenced to base relations in Table 8

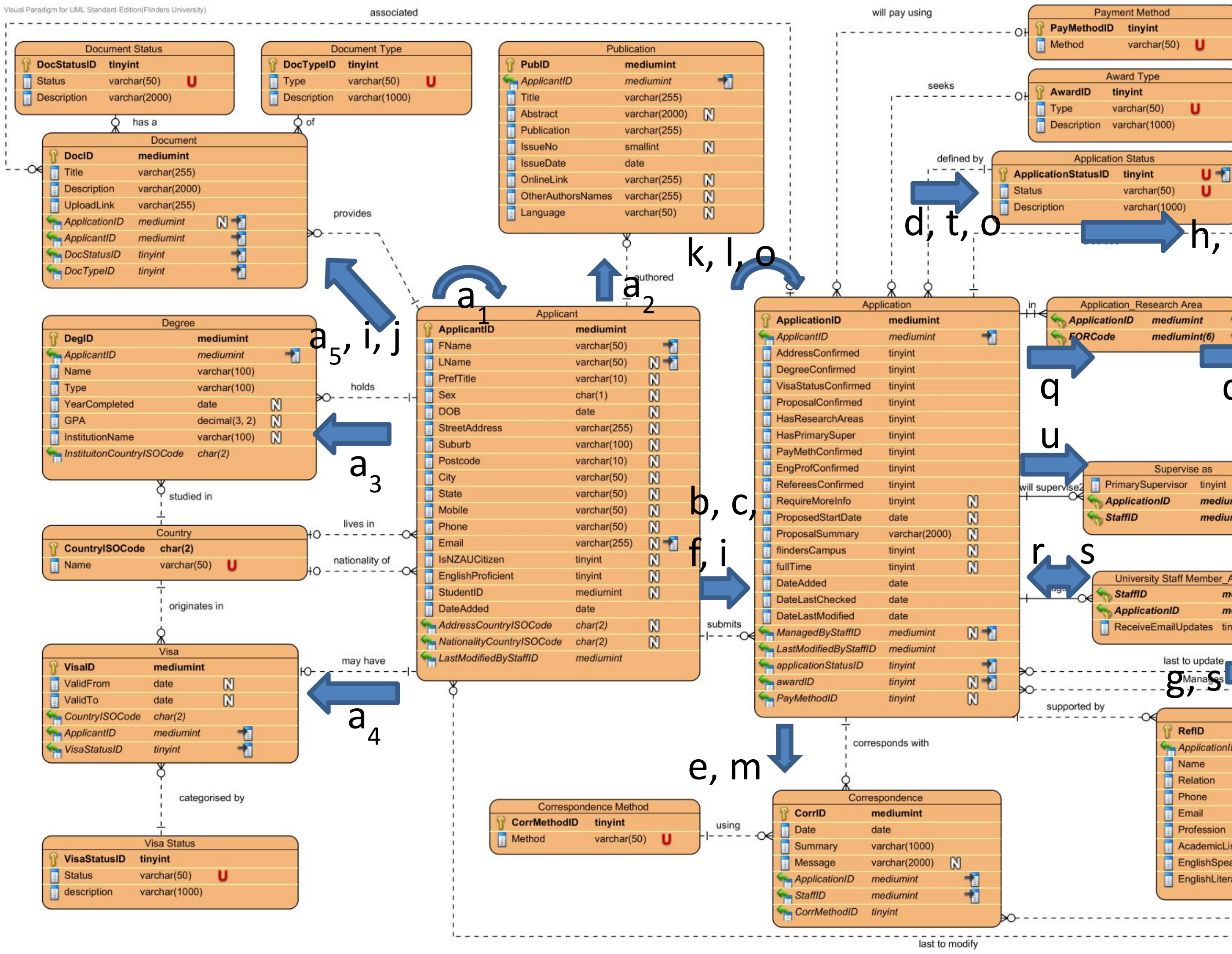


Table 9. Figure 4**Error! Reference source not found.** shows the corresponding transaction pathways.

Table 5 User transactions for all staff, including professional

u)	Look up applicant + publications + degrees + visa Status + Associated documents by applicant name
v)	Look up applicant's applications by applicant name
w)	Look up applicant's applications by applicant email
x)	Look up incomplete applications
y)	Look up all correspondences relevant to an application
z)	Create new applicant and associated application records

aa)	Look up which staff member updated an Application most recently
bb)	Check for any decision recorded about an application
cc)	Look up an existing application and attach a new standard type document to an application
dd)	Look up an existing application and attached a new exceptional type document to an application
ee)	Look up an existing application and list outstanding information (checklist).
ff)	Update the checklist to confirm that a mandatory information requirement has been met
gg)	Retrieve all on-going applications for which the user has made the most recent correspondence
hh)	Record making a decision about an application
ii)	Update the status of an application

Table 6 User transactions for academic staff

jj)	Look up, add to, and delete from own current research areas
kk)	Search for all applications in certain research areas that have been added since a certain time
ll)	Flag interest in an application

Table 7 User transactions for RHD co-ordination staff

mn	Retrieve all staff who have flagged an application, or have edited an application or applicant record most recently
nn)	Retrieve all ongoing applications
oo)	List all applications waiting for supervisor agreement

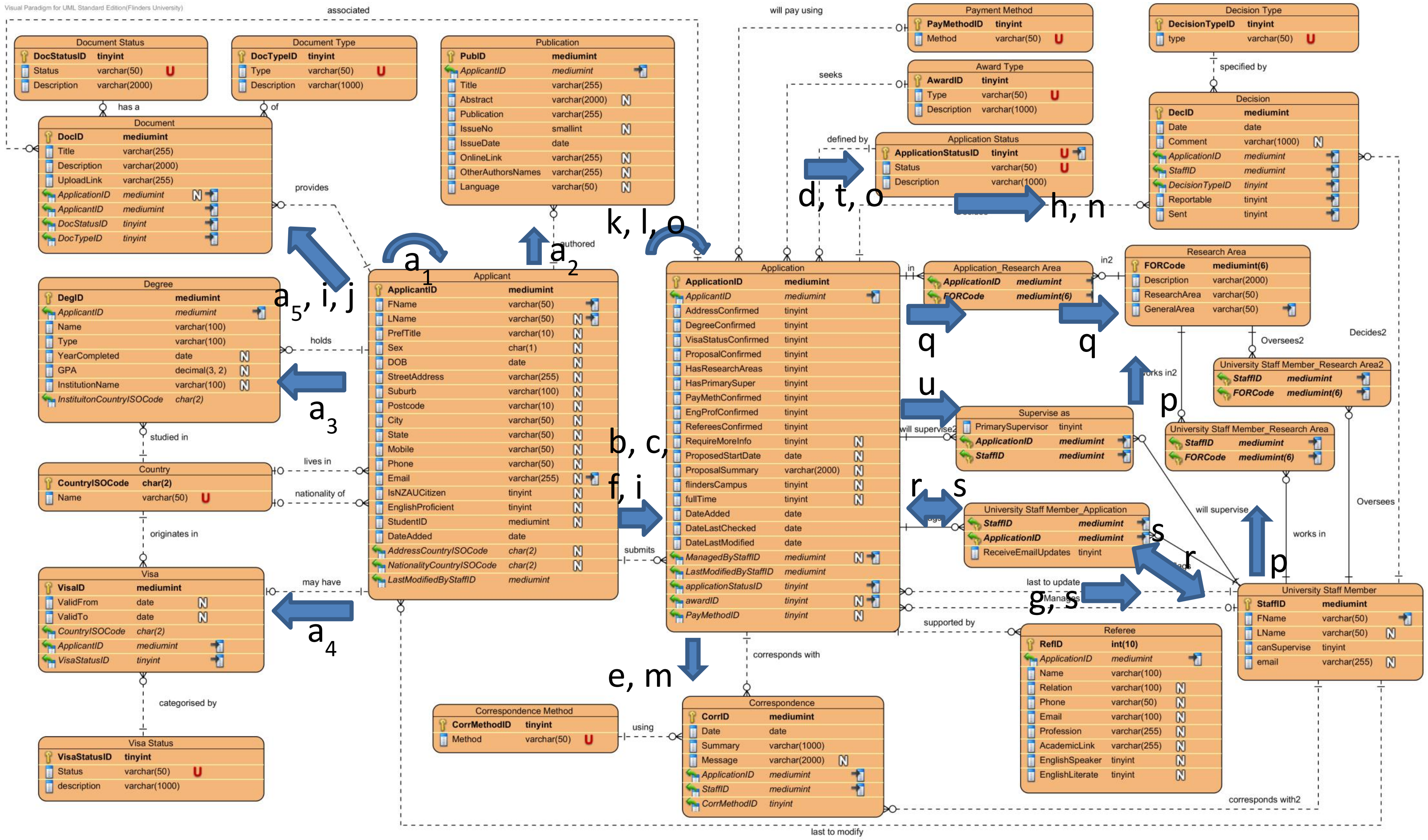


Figure 4 Transaction pathways

Table 8 Cross referencing transactions and relations. Here shown are transactions for all staff, including professional staff

	a				b				c				d				e				f				g				h				i				j				k				l				m				n				o			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Applicant		X																			X																																							
Application						X				X				X				X			X					X				X				X				X				X					X			X				X					X	
Application Status						X				X				X							X									X												X																X		
Application_Research Area						X				X											X																																							
Award Type																																																												
Correspondence																		X																																X										
Correspondence Method																		X																																										
Country																																																												
Decision																														X																							X							
Decision Type																														X																								X						
Degree		X																																																										
Document		X																															X				X																							
Document Status		X																															X				X																							
Document Type		X																															X				X																							
Payment Method																																																												
Publication		X																																																											
Referee																																																												
Research Area																																																												
Supervise as																																																												
University Staff Member																										X																																		
University Staff Member_Application																										X																																		
University Staff Member_Research Area																																																												
University Staff Member_Research Area2																																																												
Visa		X																																																											
Visa Status		X																																																											

Table 9 Cross-referencing transactions and relations. Shown here are the transactions for academic and RHD co-ordination staff

	p				q				r				s				t				u			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Applicant														X										
Application						X				X				X				X				X		
Application Status						X												X						
Application_Research Area						X																		
Award Type																								
Correspondence																								
Correspondence Method																								
Country																								
Decision																								
Decision Type																								
Degree																								
Document																								
Document Status																								
Document Type																								
Payment Method																								
Publication																								
Referee																								
Research Area						X																		
Supervise as																						X		
University Staff Member		X												X										
University Staff Member_Application									X	X	X	X		X										
University Staff Member_Research Area	X	X	X	X																				
University Staff Member_Research Area2	X	X	X	X																				
Visa																								
Visa Status																								

Analysis of the database has revealed that nearly all transactions involve the Applicant and Applications relations, as these represent the core functionality of the database.

4.2.2 Select file organisations

The file organisations are grouped by storage engines in MySQL (Oracle Inc. 2014c). The default storage engine in MySQL 5.1 is called MyISAM. The MyISAM storage engine does not provide sufficient features to help ensure the integrity of a database. For instance, it cannot enforce foreign key constraints, nor is it transaction-safe, thus cannot be relied upon to support multiple concurrent users (Oracle Inc. 2014c). InnoDB storage engine provides the required functionality for all relations. MySQL supports other storage engines, but they are all designed for specific cases that do not exist in the RHD database. Hence, we use InnoDB as the storage engine for all tables.

4.2.3 Select indices

By Default MySQL places indices on the primary key (a clustered index for the InnoDB storage engine used here (Oracle Inc. 2014d)), these are also not null enabling fast queries.

Additional Indices have also been placed on the foreign keys of

- Primary Relations: application, applicant, university staff member and Research area
  To enable fast joining between often joined relations
- Status and Type look-up Relations: Application Status, Document Status, Visa Status, Document Type and Decision Type
  To assist in common transactions

Indexes have also been placed on the first and last names and emails of applicants and staff members as these will be the primary entry into the database relations, that is, all quires of the database are expected to start by searching for an applicants or staff members name or email.

4.2.4 Estimate disk space requirements

Since all tables are set to use the INNODB storage engine, a clustered index is used as part of engine. This means that the records are physically stored (clustered) in a b-tree based on the index (left as the default primary key of each table). Each row (node of the b-tree) is then stored in a compact format (the default) reducing table space at the expense of some CPU overhead. It is also assumed that all characters are stored using the latin1 character Set with the latin1_swedish_ci. Collation (the MySQL INNODB engine defaults).[3]

As such each row has

- 1 byte per TinyInt(lookup PKs), 2bytes per smallInt (postcode) 3bytes per mediumInt (main table PKs)
- ~5 bytes per index (the header), hard to gauge MySQL documentation is rather unspecific
- 4 bytes per decimal (GPA)
- CEILING(N/8) bytes for N null columns in the row
- L+1 bytes per L length of characters used in a varchar (as all varchars used are less than 255 so 1 byte to store the length and use the latin1 Set uses 1 byte per Character)
- 6+7 bytes for the transaction ID and roll pointer fields.
- 1 or 2 bytes per non null header (2 if "if part of the column is stored externally in overflow pages or the maximum length exceeds 255 bytes and the actual length exceeds 127 bytes")

This enables the row size estimates to be calculated in the following way, i.e. for Applicant:

Date type	Estimated size in bytes
varchars	6*(50+1)+(10+1)+2*(255+1)+(100+1)+
booleans	3*1+
chars	2*1+
smallint	1*2+

[3] http://dev.mysql.com/doc/refman/5.5/en/storage-requirements.html

mediumInt	3*3+
date	1*3+
header overheads	6+7+2
TOTAL	964

Plus an additional 20 bytes (5 for each of the three indexes)

Table Name	Maximum possible size per row (bytes)	Expected max Size per row (bytes) (1/5max)*	Expected average size per filled row (bytes) (~1/9max)**
Applicant	965	196	108
Application	2060	416	230
Correspondence	3035	610	338
Decision	1038	210	116
Degree	323	68	38
Document	2542	512	284
Publication	3093	622	344
Referee	876	178	98
ResearchArea	2111	426	236
University Staff Member	367	76	42
Visa	21	21	21
Application_ Research_Area	10	10	10
Supervise As	10	10	10
University Staff Member _Applicaiton	11	11	11
University Staff Member Research Area	10	10	10
University Staff Member Research Area 2	10	10	10
Document Status	2054	414	230
Document Type	2054	414	230
Country	53	53	53
Visa Status	12	12	8
Correspondence Method	52	52	40
Payment Method	52	52	40
Award Type	1054	214	118
Application Status	1054	214	118
Decision Type	52	52	40

* and ** based on the assumption that names will use at most 10 characters with most names around 7 emails max around 45 and average around 20. With Join tables will be the max for each row.

Please note that the majority of these assumptions to create the following table are purely speculative and should be used with caution.

These values have been calculated using:
Data size = (Avg %complete by size * Expected bytes per average row * applicants per year / 1024)

Index size = (5bytes *indexNos per row * applicants per year)

Where applicants per year = 1040, 1024 is to get the calculation into Kb and then each rounded up to the nearest 16kb as the InnoDB storage engine operates in 16kb index and data chunks.

Applicants & Applications

Table Name	Avg %complete by size per applicant	Data Increase per year (kB)	Index increase per year (kB)	Total increase per year (kB)
Applicant	90% (some repeated and most simple details are included or will be devised through follow up correspondences)	112	32	144
Application	70% (no proposal)	176	32	208
Correspondence	700% (7)	2416	144	2560
Decision	1000% (7 correspondeces + 3 decisions about application)	1184	208	1392
Degree	50% (most will hold 1 degree but not specify were it was attained etc.)	32	32	64
Document	250% (2-4 documents per application)	736	64	800
Publication	5% (few will have publictions)	32	32	64
Referee	120% (most with have 2 referees but they will be only 60% complete each)	128	48	176
Visa	20%	32	32	64
Application_ Research_Area	125%	736	64	800
Supervise As	5% (most will not make it to this stage)	32	32	64
University Staff Member _Applicaiton	15% (only a few will be flagged)	128	48	176
TOTAL				5648 Kb

Staff (should be relatively constant, depending on staff members, probably tripling 10 years)

Table Name	Avg %complete by size per Staff	Data Increase per year (kB)	Index increase per year (kB)	Total increase per year (kB)
University Staff Member	100%	16	16	32
University Staff Member Research Area	400 (staff works in 4 areas)	16	16	32
University Staff Member Research Area 2	1/50 (only 1 in 50 will manage a research area)	16	16	32
TOTAL				96

Lookup relations calculated using mySQL (should be constant)

Table Name	Rows per relation	Size in KB
ResearchArea	1000	256 (240+16)*
Document Status	4	32 (16+16)
Document Type	15	32 (16+16)
Country	249	32 (16+16)
Visa Status	5	32 (16+16)
Correspondence Method	4	32 (16+16)
Payment Method	4	32 (16+16)
Award Type	4	32 (16+16)
Application Status	4	48 (32+16)
Decision Type	12	32 (16+16)
TOTAL		560kb

* calculated manually

Calculated using the following SQL, attined from http://dba.stackexchange.com/questions/46069/how-to-estimate-predict-data-size-and-index-size-of-a-table-in-mysql

```
SELECT
table_name,
    CONCAT(FORMAT(DAT/POWER(1024,pw1),2),' ',SUBSTR(units,pw1*2+1,2))
DATSIZE,
    CONCAT(FORMAT(NDX/POWER(1024,pw2),2),' ',SUBSTR(units,pw2*2+1,2))
NDXSIZE,
    CONCAT(FORMAT(TBL/POWER(1024,pw3),2),' ',SUBSTR(units,pw3*2+1,2))
TBLSIZE
FROM
(
    SELECT table_name,DAT,NDX,TBL,IF(px>4,4,px) pw1,IF(py>4,4,py)
pw2,IF(pz>4,4,pz) pw3
    FROM
    (
        SELECT table_name,data_length DAT,index_length
NDX,data_length+index_length TBL,
        FLOOR(LOG(IF(data_length=0,1,data_length))/LOG(1024)) px,
        FLOOR(LOG(IF(index_length=0,1,index_length))/LOG(1024)) py,
        FLOOR(LOG(data_length+index_length)/LOG(1024)) pz
        FROM information_schema.tables
        WHERE table_schema='dean0109'
    ) AA
) A,(SELECT 'B KBMBGBTB' units) B;
```

In summary the database is expected to grow at around 6Mb per year with staff and lookup table sizes and constants remaining relatively the same size from year to year. However it should be added this does not include added documents such as PDFs or image/document scans that will outway the entire size of database by several orders of magnitude every year. Though since little is known about what is presented by applicants and also what is deemed to be worth keeping is also unknown no estimation of these more pressing space requirements have been made at this stage.

4.3 Design user views

The database has four possible views each inheriting the previous view, as outlined by the initial requirements documentation.

These are

- Views for all staff, including professional, academic and RHD staff
    - Show all ongoing applications for which the current user staff member has an involvement. The definition of involvement includes playing a supervision role, having flagged the application, or being the staff member to most recently modify an application. The columns of this will be selected to allow the users to identify the application and applicant and the role they are playing (e.g. primary supervisor etc), and allow them to quickly determine what the next stage of developing the application.
- An ‘academic staff view’
    - A view for academic staff to list all recently added applicationsthat are in research areas that the current user has registered as working in.
- Views for RHD Co-ordination staff:
    - List all the ongoing applications and any staff member that has registered an involvement.
    - List all ongoing applications that currently haven’t had a primary supervisor assigned.

4.4 Design security mechanisms

We have adopted a light-touch to security design in this RHD database.

As the RHD application process is not strictly defined, we want the database to support a diversity of workflows. This includes fostering volunteerism by allowing any staff member to take ownership of applications.

Hence, we grant all staff members have the privilege to SELECT, UPDATE and INSERT on all tables in the database, save for the following exceptions.

Only database administrators are granted the privilege to UPDATE and INSERT to the `University Staff Member` table.

There are areas of the database that ought not to be modified without the consent of certain others. For instance, no staff members should be registered as supervising an RHD application without their own consent. The other, similarly sensitive areas are: a staff member's research areas and their oversight responsibilities for research areas; and which applications they have flagged.

To address this issue, we log all changes to the related tables so that the affected staff can be emailed summaries of such changes. This design has the benefit of allowing any user to make changes to these areas so potentially sharing widely the work of keeping the database up-to-date. In addition, those staff directly affected by such changes can be automatically forwarded notifications of those changes. By sharing more widely the responsibility for keeping this database up-to-date, we can minimise the extra time academic staff in particular need to spend on the system.

Deletion may only be performed by RHD Admin and this is assumed to occur very rarely.

4.5 Introduce controlled redundancy if necessary

One form of redundancy that we already have in our design is the checklist feature. This is the set of ten Boolean columns in the Application relation: AddressConfirmed, DegreeConfirmed, VisaStatusConfirmed, ProposalConfirmed, HasResearchAreas, HasPrimarySuper, PayMethodConfirmed, EngProfConfirmed, RefereesConfirmed and RequiresMoreInfo. All these values are derivable from other primary values in the schema. But we summarise them here to: allow the human users of the system to control theses values in a flexible way; and, less importantly, to increase the performance of the system so that these values do not need to be recalculated frequently.

We have decided to duplicate the Applicant(Email) attribute into the Application table. As email is the most common medium for communicating with applicants, it will almost always be present when an email is received with new information to update an application, and conversely when working on an application without being prompted by an email, the address should be retrieved by the system to help the user identify the applicant and send a new email to them. The email address can also be useful for users to search their email clients' for messages sent and received about an application. Of course, care will have to be taken to ensure that this redundant information doesn't become inconsistent. This is an example of duplicating a non-key attribute across a 1:* relationship, which we expect to reduce the number of join operations on the Applicant, Application tables significantly. Also, email addresses aren't likely to change very often, so keeping this information consistent across multiple redundant copies should not be too much work for a database system.

We also have considered other types of performance improvements by introducing controlled redundancy. We could combine one-to-one relationships, but we do not have any such relationships in our design. We could duplicate foreign key attributes across one-to-many relationships to reduce joins, but we do not have foreign key attributes used in joins sufficiently frequently to make this worthwhile. We could duplicate attributes in many-to-many relationships to reduce joins, but the many-to-many tables we have do not have many attributes. We considered introducing repeating groups, but decided there were no opportunities to do so in our design.

Creating extract tables is another way of increasing performance by introducing redundancy. We decided this would be best done in future once the system has been in use for a while. This would enable an accurate understanding of which frequently used reports slow the system down, and of those, for which it is appropriate to use potentially somewhat out-of-date data.

Another strategy to increase performance is partitioning relations. The most obvious application of this strategy to our design is to perform a horizontal partition on the Application table, separating all ongoing applications from completed applications. As almost all lookups on that table are expected to be on ongoing relations, rather than slow these down by scanning through one table that contains all applications ever created, it could search over a far smaller partition of only the ongoing relations.

Another way of achieving the same effect would be to create another copy of the Application table, and store all ongoing applications in the primary table, and move all complete applications to the secondary copy table. As this is not a common feature of database management systems.

4.6 Create SQL scripts for data definition

4.6.1 Create base tables, constraints and indexes

```
-- ---------------------------------------------------------------------------
--
-- Create or re-create the RHD tables, constraints and indexes
--
-- ---------------------------------------------------------------------------

-- ---------------------------------------------------------------------------
-- Delete existing tables

SELECT "Dropping existing FK constraints" ;
ALTER TABLE Document DROP FOREIGN KEY associated;
ALTER TABLE Document DROP FOREIGN KEY provides;
ALTER TABLE Decision DROP FOREIGN KEY `specified by`;
ALTER TABLE Visa DROP FOREIGN KEY `originates in`;
ALTER TABLE Degree DROP FOREIGN KEY `studied in`;
ALTER TABLE Applicant DROP FOREIGN KEY `lives in`;
ALTER TABLE Applicant DROP FOREIGN KEY `nationality of`;
ALTER TABLE Application DROP FOREIGN KEY `will pay using`;
ALTER TABLE Application DROP FOREIGN KEY submits;
ALTER TABLE `Supervise as` DROP FOREIGN KEY `will supervise2`;
ALTER TABLE `Supervise as` DROP FOREIGN KEY `will supervise`;
ALTER TABLE `Application_Research Area` DROP FOREIGN KEY `in`;
ALTER TABLE `Application_Research Area` DROP FOREIGN KEY in2;
ALTER TABLE `University Staff Member_Research Area` DROP FOREIGN KEY `works in`;
ALTER TABLE `University Staff Member_Research Area` DROP FOREIGN KEY
            `works in2`;
ALTER TABLE `University Staff Member_Research Area2` DROP FOREIGN KEY Oversees;
ALTER TABLE `University Staff Member_Research Area2` DROP FOREIGN KEY Oversees2;
ALTER TABLE Decision DROP FOREIGN KEY Decides;
ALTER TABLE Decision DROP FOREIGN KEY Decides2;
ALTER TABLE Visa DROP FOREIGN KEY `categorised by`;
ALTER TABLE Application DROP FOREIGN KEY `defined by`;
ALTER TABLE Document DROP FOREIGN KEY `has a`;
ALTER TABLE `University Staff Member_Application` DROP FOREIGN KEY flags;
ALTER TABLE `University Staff Member_Application` DROP FOREIGN KEY flags2;
ALTER TABLE Referee DROP FOREIGN KEY `supported by`;
ALTER TABLE Correspondence DROP FOREIGN KEY `corresponds with`;
ALTER TABLE Correspondence DROP FOREIGN KEY `corresponds with2`;
ALTER TABLE Document DROP FOREIGN KEY `of`;
ALTER TABLE Application DROP FOREIGN KEY Manages;
ALTER TABLE Degree DROP FOREIGN KEY holds;
ALTER TABLE Publication DROP FOREIGN KEY authored;
ALTER TABLE Visa DROP FOREIGN KEY `may have`;
ALTER TABLE Application DROP FOREIGN KEY seeks;
ALTER TABLE Application DROP FOREIGN KEY `last to update`;
ALTER TABLE Applicant DROP FOREIGN KEY `last to modify`;
ALTER TABLE Correspondence DROP FOREIGN KEY `using`;

SELECT "Dropping tables" ;
DROP TABLE IF EXISTS `Decision Type`;
DROP TABLE IF EXISTS `Payment Method`;
DROP TABLE IF EXISTS Application;
DROP TABLE IF EXISTS `Document Status`;
DROP TABLE IF EXISTS Publication;
DROP TABLE IF EXISTS Degree;
DROP TABLE IF EXISTS Referee;
DROP TABLE IF EXISTS Document;
DROP TABLE IF EXISTS Applicant;
DROP TABLE IF EXISTS Visa;
DROP TABLE IF EXISTS `Visa Status`;
DROP TABLE IF EXISTS Correspondence;
DROP TABLE IF EXISTS Decision;
DROP TABLE IF EXISTS `Research Area`;
DROP TABLE IF EXISTS `University Staff Member`;
DROP TABLE IF EXISTS `Correspondence Method`;
DROP TABLE IF EXISTS `Application Status`;
DROP TABLE IF EXISTS `Document Type`;
DROP TABLE IF EXISTS `Award Type`;
DROP TABLE IF EXISTS Country;
DROP TABLE IF EXISTS `Supervise as`;
DROP TABLE IF EXISTS `Application_Research Area`;
DROP TABLE IF EXISTS `University Staff Member_Research Area`;
DROP TABLE IF EXISTS `University Staff Member_Research Area2`;
DROP TABLE IF EXISTS `University Staff Member_Application`;

-- ----------------------------------------------------------------------------
-- Create tables
SELECT "Creating tables" ;

CREATE TABLE `Decision Type` (
  DecisionTypeID mediumint NOT NULL AUTO_INCREMENT,
  type varchar(50) NOT NULL UNIQUE comment 'the type of decision/comment made',
  PRIMARY KEY (DecisionTypeID)
) comment='the possible types of decisions/comments that can be made'
ENGINE=InnoDB;

CREATE TABLE `Payment Method` (
  PayMethodID mediumint NOT NULL AUTO_INCREMENT,
  Method varchar(50) NOT NULL UNIQUE comment 'Method of payment, e.g. scholarship,
letter of financial support etc.',
  PRIMARY KEY (PayMethodID)
)
comment='the possible payment methods of a Research higher degree'
ENGINE=InnoDB;

CREATE TABLE Application (
  ApplicationID mediumint NOT NULL AUTO_INCREMENT,
  ApplicantEmail varchar (100)
    comment 'Denormalised column to improve retrieval of applications via email
address',
  ApplicantID mediumint NOT NULL
    comment 'the ID of the applicant who proposed this application',
  AddressConfirmed tinyint(1) NOT NULL comment 'All contact details appear valid',
  DegreeConfirmed tinyint(1) NOT NULL
    comment 'The degree is a recognised degree of the institution',
  VisaStatusConfirmed tinyint(1) NOT NULL
    comment 'The visa status is backed by an official document',
  ProposalConfirmed tinyint(1) NOT NULL
    comment 'The proposal is contains appropriate detail',
  HasResearchAreas tinyint(1) NOT NULL
    comment 'Has nominated research areas relevant to the proposal',
  HasPrimarySuper tinyint(1) NOT NULL
    comment 'Has the required number of supervisors',
  PayMethConfirmed tinyint(1) NOT NULL
    comment 'The payment method is backed by an official document',
  EngProfConfirmed tinyint(1) NOT NULL
    comment 'The applicant has some level of English literacy',
  RefereesConfirmed tinyint(1) NOT NULL
    comment 'The referees details appear to be correct',
  RequireMoreInfo tinyint(1),
  ProposedStartDate date
    comment 'The date the applicant prefers to start the RHD (Entered as 1/1/## for
S1 and 1/7/## for S2)',
  ProposalSummary varchar(2000)
    comment 'What the proposal is about',
  flindersCampus tinyint(1)
    comment 'the applicant wants to complete the degree a main campus',
  fullTime tinyint(1) comment 'the applicant wants to undergo the degree full
time',
  DateAdded date NOT NULL comment 'the date the application was added',
  DateLastChecked date NOT NULL
```

```
    comment 'the date the application was last checked',
  DateLastModified date NOT NULL
    comment 'the date the application was last modified',
  ManagedByStaffID mediumint
    comment 'The staff ID of the staff member who has been personally assigned to
manage this application (since it may not be used it is nullable)',
  LastModifiedByStaffID mediumint NOT NULL
    comment 'the staff member ID of the last person to modify the application (all
modifications are recorded in the decision table)',
  applicationStatusID mediumint NOT NULL,
  awardID mediumint,
  PayMethodID mediumint,
  PRIMARY KEY (ApplicationID),
  INDEX (ApplicantID),
  INDEX (applicationStatusID),
  INDEX (awardID),
  INDEX (ApplicantEmail)
)
comment='Holds the application details including a checklist of recorded
information and applicaiton stage'
ENGINE=InnoDB;

CREATE TABLE `Document Status` (
  DocStatusID mediumint NOT NULL AUTO_INCREMENT,
  Status varchar(50) NOT NULL UNIQUE comment 'Official and translation status of a
document associated to an Applicant',
  Description varchar(2000) NOT NULL comment 'the details and implication of this
status',
  PRIMARY KEY (DocStatusID)
)
comment='the possible statuses of a document'
ENGINE=InnoDB;

CREATE TABLE Publication (
  PubID mediumint NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely
identifies the publication',
  ApplicantID mediumint NOT NULL comment 'the ID of the applicant who authored the
publication',
  Title varchar(255) NOT NULL comment 'The title of the publication',
  Abstract varchar(2000) comment 'A abstract/description of the publication',
  Publication varchar(255) NOT NULL comment 'The journal/magazine publisher ',
  IssueNo mediumint comment 'The issue/edition number of the publication',
  IssueDate date NOT NULL comment 'The date the publication was issued',
  OnlineLink varchar(255) comment 'An online link to the publication',
  OtherAuthorsNames varchar(255) comment 'Other authors of the publication',
  Language varchar(50) comment 'language of the publication',
  PRIMARY KEY (PubID))
ENGINE=InnoDB;

CREATE TABLE Degree (
  DegID mediumint NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely
identifies the degree',
  ApplicantID mediumint NOT NULL comment 'the ID of the applicant who holds this
degree',
  Name varchar(100) NOT NULL comment 'The title of the degree',
  Type varchar(100) NOT NULL comment 'The type of the degree  -Â Â Could add
specific types',
  YearCompleted date comment 'The year the degree was completed or will be
completed',
  GPA mediumint comment 'The GPA of the degree',
  InstitutionName varchar(100) comment 'The name of the institution',
  InstitutionCountryISOCode char(2) NOT NULL comment 'the country the institution
is based in',
  PRIMARY KEY (DegID),
  INDEX (ApplicantID)
)
comment='Any Degrees already held by the applicant'
ENGINE=InnoDB;
```

```
CREATE TABLE Referee (
  RefID mediumint NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely
identifies the referee supporting an application',
  ApplicationID mediumint NOT NULL,
  Name varchar(100) NOT NULL comment 'The full name of the referee',
  Relation varchar(100) comment 'The referees relation to the applicant',
  Phone varchar(50) comment 'The referees phone number',
  Email varchar(100) comment 'The referees email address',
  Profession varchar(255) comment 'The referees profession',
  AcademicLink varchar(255) comment 'The referees professional page (linked in or
university)',
  EnglishSpeaker tinyint(1) comment 'If the Referee can speak English',
  EnglishLiterate tinyint(1) comment 'If the Referee can read and write in
English',
  PRIMARY KEY (RefID)
)
comment='A referee for a application'
ENGINE=InnoDB;

CREATE TABLE Document (
  DocID mediumint NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely
identifies the document',
  Title varchar(254) NOT NULL comment 'The title of the document',
  Description varchar(2000) NOT NULL comment 'A specific summary related to the
document i.e. valid till 2015 etc.',
  UploadLink varchar(254) NOT NULL comment 'An link to a version uploaded and
stored on the university servers',
  ApplicationID mediumint,
  ApplicantID mediumint NOT NULL comment 'the ID of the applicant who provided this
document',
  DocStatusID mediumint NOT NULL,
  DocTypeID mediumint NOT NULL,
  PRIMARY KEY (DocID),
  INDEX (ApplicationID),
  INDEX (ApplicantID),
  INDEX (DocStatusID),
  INDEX (DocTypeID)
)
comment='Links to any relevant documents along with descriptions,
  types and statuses.'
ENGINE=InnoDB;

CREATE TABLE Applicant (
  ApplicantID mediumint NOT NULL AUTO_INCREMENT
    comment 'The primary key that uniquely identifies the applicant',
  FName varchar(50) NOT NULL comment 'First name',
  LName varchar(50) comment 'Last name',
  PrefTitle varchar(10) comment 'Title Mr,
  Mrs,
  Miss,
  Dr. *',
  Sex tinyint(1) comment 'The sex of the applicant',
  DOB date comment 'Date of birth',
  StreetAddress varchar(255) comment 'Residence number and street of residence',
  Suburb varchar(100) comment 'The suburb of residence',
  Postcode mediumint comment 'The postcode of residence',
  City varchar(50) comment 'The city or town of residence',
  State varchar(50) comment 'The State of residence',
  Mobile varchar(50) comment 'Mobile phone number',
  Phone varchar(50) comment 'Landline phone number',
  Email varchar(100) comment 'The email address of the applicant',
  IsNZAUCitizen tinyint(1) comment 'Is a new Zealand or Australian citizen, a check
to see if visa information is required ***',
  EnglishProficient tinyint(1) comment 'English ability',
  StudentID mediumint comment 'The flinders university student id if they are or
have been enrolled at flinders university',
  DateAdded date NOT NULL comment 'The date the applicant was added to the system',
  AddressCountryISOCode char(2),
  NationalityCountryISOCode char(2),
  LastModifiedByStaffID mediumint NOT NULL comment 'the staff member ID of the last
person to modify the applicant (all modifications are recorded in the decision
table)',
  PRIMARY KEY (ApplicantID),
  INDEX (FName),
  INDEX (LName),
  INDEX (Email)
)
comment='Holds the applicant specific details'
ENGINE=InnoDB;

CREATE TABLE Visa (
  VisaID mediumint NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely
identifies the visa',
  ValidFrom date comment 'When the visa is valid from',
  ValidTo date comment 'When the visa is valid to',
  CountryISOCode char(2) NOT NULL comment 'the applicant country, the visa is
granted to',
  ApplicantID mediumint NOT NULL comment 'the ID of the applicant who holds or may
hold this visa',
  VisaStatusID mediumint NOT NULL,
  PRIMARY KEY (VisaID),
  INDEX (ApplicantID),
  INDEX (VisaStatusID)
)
comment='The applicants visa details'
ENGINE=InnoDB;

CREATE TABLE `Visa Status` (
  VisaStatusID mediumint NOT NULL AUTO_INCREMENT,
  Status varchar(50) NOT NULL UNIQUE comment 'the status of the visa application',
  description varchar(1000) NOT NULL comment 'a description of the status of the
visa',
  PRIMARY KEY (VisaStatusID)
)
comment='the possible statuses of the visa application'
ENGINE=InnoDB;

CREATE TABLE Correspondence (
  CorrID mediumint NOT NULL AUTO_INCREMENT,
  `Date` date NOT NULL comment 'The date the correspondence was made/received',
  Summary varchar(1000) NOT NULL comment 'A small summary of the Correspondence',
  Message varchar(2000) comment 'The actual message contained in the
correspondence',
  ApplicationID mediumint NOT NULL comment 'the application ID the correspondence
is in relation to',
  StaffID mediumint NOT NULL comment 'the staff ID of the staff member the
correspondence is to/from',
  CorrMethodID mediumint NOT NULL,
  PRIMARY KEY (CorrID),
  INDEX (ApplicationID),
  INDEX (StaffID)
)
comment='Correspondence between the Applicant and University Staff Member'
ENGINE=InnoDB;

CREATE TABLE Decision (
  DecID mediumint NOT NULL AUTO_INCREMENT comment 'The primary key that uniquely
identifies the Decision/comment made',
  `Date` date NOT NULL comment 'The date the decision was made on',
  Comment varchar(1000) comment 'Extra information about the decision',
  ApplicationID mediumint comment 'the id of the application this decision is made
with regards to, if this is a decision associated with an application',
  StaffID mediumint NOT NULL comment 'the staff ID of the staff member who made
this decision/comment',
  DecisionTypeID mediumint NOT NULL,
  Reportable tinyint(1) NOT NULL comment 'a boolean that is automatically ticked if
the change is deemed reportable (status changes request filled etc.)',
  Sent tinyint(1) comment 'a boolean to check if the related email has been sent',
  PRIMARY KEY (DecID),
  INDEX (ApplicationID),
  INDEX (StaffID),
  INDEX (DecisionTypeID),
  INDEX (Reportable),
  INDEX (Sent)
)
comment='The decision/comment made for an application by a RHD staff member'
ENGINE=InnoDB;

CREATE TABLE `Research Area` (
  FORCode mediumint(6) NOT NULL AUTO_INCREMENT comment 'The Australian Field Of
Research (FOR) code,
  primary key,
  that uniquely identifies the Research area',
  Description varchar(2000) NOT NULL comment 'A small text description of the FOR
i.e. 2201',
  ResearchArea varchar(50) NOT NULL comment 'The FOR title; i.e. Applied Ethics',
  GeneralArea varchar(50) NOT NULL comment 'The general area of the FOR',
  PRIMARY KEY (FORCode),
  INDEX (GeneralArea))
ENGINE=InnoDB;

CREATE TABLE `University Staff Member` (
  StaffID mediumint NOT NULL AUTO_INCREMENT comment 'The flinders uni staff ID
number,
  the primary key that uniquely identifies the staff member',
  FName varchar(50) NOT NULL comment 'The last name of the staff member',
  LName varchar(50) comment 'The first name of the staff member',
  canSupervise tinyint(1) NOT NULL comment 'if the staff member is able supervise a
RHD applicant',
  email varchar(100) NOT NULL,
  PRIMARY KEY (StaffID),
  INDEX (FName)
)
comment='a university staff member who may be able to supervise a application'
ENGINE=InnoDB;

CREATE TABLE `Correspondence Method` (
  CorrMethodID mediumint NOT NULL AUTO_INCREMENT,
  Method varchar(50) NOT NULL UNIQUE comment 'the method of correspondence',
  PRIMARY KEY (CorrMethodID))
ENGINE=InnoDB;

CREATE TABLE `Application Status` (
  ApplicationStatusID mediumint NOT NULL AUTO_INCREMENT,
  Status varchar(50) NOT NULL UNIQUE comment 'the name of the status',
  Description varchar(1000) NOT NULL comment 'a full description of the status',
  PRIMARY KEY (ApplicationStatusID),
  UNIQUE INDEX (ApplicationStatusID)
)
comment='the possible application statuses of an application'
ENGINE=InnoDB;

CREATE TABLE `Document Type` (
  DocTypeID mediumint NOT NULL AUTO_INCREMENT,
  Type varchar(50) NOT NULL UNIQUE comment 'the type of document eg. Publication,
  visa etc.',
  Description varchar(1000) NOT NULL comment 'a full description of the type of the
document',
  PRIMARY KEY (DocTypeID)
)
comment='the possible types of a document'
ENGINE=InnoDB;
```

```
CREATE TABLE `Award Type` (
  AwardID mediumint NOT NULL AUTO_INCREMENT,
  Type varchar(50) NOT NULL UNIQUE comment 'the type of award sought by the
applicant',
  Description varchar(1000) NOT NULL comment 'a full description of the award
type',
  PRIMARY KEY (AwardID)
)
comment='the possible award types (degrees) sought by an application'
ENGINE=InnoDB;

CREATE TABLE Country (
  CountryISOCode char(2) NOT NULL
    comment 'the country corresponding ISO 3166-1 alpha-2 code',
  Name varchar(50) NOT NULL UNIQUE comment 'the full name of the country',
  PRIMARY KEY (CountryISOCode)
)
comment='a list of countries for reuse in nationality,
  institution country,
  address and visa country'
ENGINE=InnoDB;

CREATE TABLE `Supervise as` (
  PrimarySupervisor tinyint(1) NOT NULL comment 'if the supervisor is a primary',
  ApplicationID mediumint NOT NULL comment 'the application ID of the application
the staff member will supervise',
  StaffID mediumint NOT NULL comment 'the staff ID of the staff member who will
supervise the applicaiton',
  PRIMARY KEY (ApplicationID,
  StaffID),
  INDEX (ApplicationID),
  INDEX (StaffID)
)
comment='the staff members who have agreed to supervise an application'
ENGINE=InnoDB;

CREATE TABLE `Application_Research Area` (
  ApplicationID mediumint NOT NULL comment 'the ID of the application',
  FORCode mediumint(6) NOT NULL comment 'the FORCode research area the applicant
states they want to study in',
  PRIMARY KEY (ApplicationID, FORCode),
  INDEX (ApplicationID),
  INDEX (FORCode)
)
comment='The application and the research area they are looking to study in'
ENGINE=InnoDB;

CREATE TABLE `University Staff Member_Research Area` (
  StaffID mediumint NOT NULL comment 'the staff ID of the staff member who works in
the research area',
  FORCode mediumint(6) NOT NULL comment 'the field of research that the staff
member works in',
  PRIMARY KEY (StaffID, FORCode),
  INDEX (StaffID),
  INDEX (FORCode)
)
comment='the research area the staff member states they work in'
ENGINE=InnoDB;

CREATE TABLE `University Staff Member_Research Area2` (
  StaffID mediumint NOT NULL comment 'the staff ID of the staff member who oversees
the research area',
  FORCode mediumint(6) NOT NULL comment 'the FORCode of the research area that the
staff member oversees',
  PRIMARY KEY (StaffID, FORCode),
  INDEX (StaffID),
  INDEX (FORCode)
)
```

```
comment='the research areas that a staff member oversees (can be more than one
staff member per area)'
ENGINE=InnoDB;

CREATE TABLE `University Staff Member_Application` (
  StaffID mediumint NOT NULL comment 'the staff ID of the staff member who flagged
the application',
  ApplicationID mediumint NOT NULL comment 'the application the staff member has
flagged',
  ReceiveEmailUpdates tinyint(1) NOT NULL comment 'a boolean that a user can check
if they want to be alerted About an update or simply keep a reference on their
application page',
  PRIMARY KEY (StaffID,
  ApplicationID),
  INDEX (StaffID),
  INDEX (ApplicationID)
)
comment='the staff members who have flagged an application to come back and check
later'
ENGINE=InnoDB;


-- ---------------------------------------------------------------------------------
-- Add constraints and indexes
SELECT "Adding indexes and FK constraints" ;

ALTER TABLE Document
ADD INDEX associated (ApplicationID),
ADD CONSTRAINT associated FOREIGN KEY (ApplicationID)
  REFERENCES Application (ApplicationID) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Document
ADD INDEX provides (ApplicantID),
ADD CONSTRAINT provides FOREIGN KEY (ApplicantID)
  REFERENCES Applicant (ApplicantID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Decision
ADD INDEX `specified by` (DecisionTypeID),
ADD CONSTRAINT `specified by` FOREIGN KEY (DecisionTypeID)
  REFERENCES `Decision Type` (DecisionTypeID) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Visa
ADD INDEX `originates in` (CountryISOCode),
ADD CONSTRAINT `originates in` FOREIGN KEY (CountryISOCode)
  REFERENCES Country (CountryISOCode) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Degree
ADD INDEX `studied in` (InstitutionCountryISOCode),
ADD CONSTRAINT `studied in` FOREIGN KEY (InstitutionCountryISOCode)
  REFERENCES Country (CountryISOCode) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Applicant
ADD INDEX `lives in` (AddressCountryISOCode),
ADD CONSTRAINT `lives in` FOREIGN KEY (AddressCountryISOCode)
  REFERENCES Country (CountryISOCode) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Applicant
ADD INDEX `nationality of` (NationalityCountryISOCode),
ADD CONSTRAINT `nationality of` FOREIGN KEY (NationalityCountryISOCode)
  REFERENCES Country (CountryISOCode) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Application
ADD INDEX `will pay using` (PayMethodID),
ADD CONSTRAINT `will pay using` FOREIGN KEY (PayMethodID)
  REFERENCES `Payment Method` (PayMethodID)
  ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Application
```

```
ADD INDEX submits (ApplicantID),
ADD CONSTRAINT submits FOREIGN KEY (ApplicantID)
  REFERENCES Applicant (ApplicantID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE `Supervise as`
ADD INDEX `will supervise2` (ApplicationID),
ADD CONSTRAINT `will supervise2` FOREIGN KEY (ApplicationID)
  REFERENCES Application (ApplicationID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE `Supervise as`
ADD INDEX `will supervise` (StaffID),
ADD CONSTRAINT `will supervise` FOREIGN KEY (StaffID)
  REFERENCES `University Staff Member` (StaffID)
  ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE `Application_Research Area`
ADD INDEX `in` (ApplicationID),

ADD CONSTRAINT `in` FOREIGN KEY (ApplicationID)
  REFERENCES Application (ApplicationID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE `Application_Research Area`
ADD INDEX in2 (FORCode),

ADD CONSTRAINT in2 FOREIGN KEY (FORCode)
  REFERENCES `Research Area` (FORCode) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE `University Staff Member_Research Area`
ADD INDEX `works in` (StaffID),

ADD CONSTRAINT `works in` FOREIGN KEY (StaffID)
  REFERENCES `University Staff Member` (StaffID) ON UPDATE Cascade ON DELETE
Cascade;

ALTER TABLE `University Staff Member_Research Area`
ADD INDEX `works in2` (FORCode),

ADD CONSTRAINT `works in2` FOREIGN KEY (FORCode)
  REFERENCES `Research Area` (FORCode) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE `University Staff Member_Research Area2`
ADD INDEX Oversees (StaffID),

ADD CONSTRAINT Oversees FOREIGN KEY (StaffID)
  REFERENCES `University Staff Member` (StaffID) ON UPDATE Cascade ON DELETE
Restrict;

ALTER TABLE `University Staff Member_Research Area2`
ADD INDEX Oversees2 (FORCode),

ADD CONSTRAINT Oversees2 FOREIGN KEY (FORCode)
  REFERENCES `Research Area` (FORCode) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Decision
ADD INDEX Decides (ApplicationID),
ADD CONSTRAINT Decides FOREIGN KEY (ApplicationID)
  REFERENCES Application (ApplicationID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Decision
ADD INDEX Decides2 (StaffID),
ADD CONSTRAINT Decides2 FOREIGN KEY (StaffID)
  REFERENCES `University Staff Member` (StaffID)
  ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Visa
ADD INDEX `categorised by` (VisaStatusID),
ADD CONSTRAINT `categorised by` FOREIGN KEY (VisaStatusID)
  REFERENCES `Visa Status` (VisaStatusID) ON UPDATE Cascade ON DELETE Restrict;
```

```
ALTER TABLE Application
ADD INDEX `defined by` (applicationStatusID),
ADD CONSTRAINT `defined by` FOREIGN KEY (applicationStatusID)
  REFERENCES `Application Status` (ApplicationStatusID)
  ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Document
ADD INDEX `has a` (DocStatusID),
ADD CONSTRAINT `has a` FOREIGN KEY (DocStatusID)
  REFERENCES `Document Status` (DocStatusID)
  ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE `University Staff Member_Application`
ADD INDEX flags (StaffID),
ADD CONSTRAINT flags FOREIGN KEY (StaffID)
  REFERENCES `University Staff Member` (StaffID)
  ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE `University Staff Member_Application`
ADD INDEX flags2 (ApplicationID),
ADD CONSTRAINT flags2 FOREIGN KEY (ApplicationID)
  REFERENCES Application (ApplicationID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Referee
ADD INDEX `supported by` (ApplicationID),
ADD CONSTRAINT `supported by` FOREIGN KEY (ApplicationID)
  REFERENCES Application (ApplicationID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Correspondence
ADD INDEX `corresponds with` (ApplicationID),
ADD CONSTRAINT `corresponds with` FOREIGN KEY (ApplicationID)
  REFERENCES Application (ApplicationID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Correspondence
ADD INDEX `corresponds with2` (StaffID),
ADD CONSTRAINT `corresponds with2` FOREIGN KEY (StaffID)
  REFERENCES `University Staff Member` (StaffID)
  ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Document
ADD INDEX `of` (DocTypeID),
ADD CONSTRAINT `of` FOREIGN KEY (DocTypeID)
  REFERENCES `Document Type` (DocTypeID) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Application
ADD INDEX Manages (ManagedByStaffID),
ADD CONSTRAINT Manages FOREIGN KEY (ManagedByStaffID)
  REFERENCES `University Staff Member` (StaffID)
  ON UPDATE Cascade ON DELETE Set null;

ALTER TABLE Degree
ADD INDEX holds (ApplicantID),
ADD CONSTRAINT holds FOREIGN KEY (ApplicantID)
  REFERENCES Applicant (ApplicantID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Publication
ADD INDEX authored (ApplicantID),
ADD CONSTRAINT authored FOREIGN KEY (ApplicantID)
  REFERENCES Applicant (ApplicantID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Visa
ADD INDEX `may have` (ApplicantID),
ADD CONSTRAINT `may have` FOREIGN KEY (ApplicantID)
  REFERENCES Applicant (ApplicantID) ON UPDATE Cascade ON DELETE Cascade;

ALTER TABLE Application
ADD INDEX seeks (awardID),
ADD CONSTRAINT seeks FOREIGN KEY (awardID)
  REFERENCES `Award Type` (AwardID) ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Application
ADD INDEX `last to update` (LastModifiedByStaffID),
ADD CONSTRAINT `last to update` FOREIGN KEY (LastModifiedByStaffID)
  REFERENCES `University Staff Member` (StaffID)
  ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Applicant
ADD INDEX `last to modify` (LastModifiedByStaffID),
ADD CONSTRAINT `last to modify` FOREIGN KEY (LastModifiedByStaffID)
  REFERENCES `University Staff Member` (StaffID)
  ON UPDATE Cascade ON DELETE Restrict;

ALTER TABLE Correspondence
ADD INDEX `using` (CorrMethodID),
ADD CONSTRAINT `using` FOREIGN KEY (CorrMethodID)
  REFERENCES `Correspondence Method` (CorrMethodID)
  ON UPDATE Cascade ON DELETE Restrict;


-- Load up other parts
source create_views.sql
source security.sql
source create_triggers.sql
```

4.6.2 Create views

4.6.2.1 Utilities

```
SELECT 'Creating utility views';

-- A utility view with lookup values cached out
DROP VIEW IF EXISTS Application_Expanded ;
CREATE VIEW Application_Expanded AS
SELECT Appt.FName, Appt.LName, Appt.Email, App.*, AppStat.Status
FROM Application App
INNER JOIN Applicant Appt
  ON Appt.ApplicantID = App.ApplicantID
INNER JOIN `Application Status` AppStat
  ON AppStat.ApplicationStatusID = App.applicationStatusID;

DROP VIEW IF EXISTS Application_Ongoing_Expanded ;
CREATE VIEW Application_Ongoing_Expanded AS
SELECT *
FROM Application_Expanded App
WHERE App.Status LIKE 'ongoing%';

-- Friendly view of application status
DROP VIEW IF EXISTS `Application_Quickview` ;
CREATE VIEW `Application_Quickview` AS
SELECT
  FName,
  LName,
  Email,
  Status,
  AddressConfirmed,
  DegreeConfirmed,
  VisaStatusConfirmed,
  ProposalConfirmed,
  HasResearchAreas,
  HasPrimarySuper,
  PayMethConfirmed,
  EngProfConfirmed,
  RefereesConfirmed
```

```
FROM Application_Expanded
GROUP BY ApplicationID
ORDER BY LName;

-- Create views to separate primary supervisors from associate
DROP VIEW IF EXISTS `Supervise as primary` ;
CREATE VIEW `Supervise as primary` AS
SELECT *
FROM `Supervise as` AS Super
WHERE Super.PrimarySupervisor = 1;

DROP VIEW IF EXISTS `Supervise as associate` ;
CREATE VIEW `Supervise as associate` AS
SELECT *
FROM `Supervise as` AS Super
WHERE Super.PrimarySupervisor = 0;



DELIMITER $$

-- A function to simulate user switching, as we don’t have facility to do this
-- for real on the university system
-- +--------+--------+----------+--------------+-------------------------------+
-- |StaffID | FName  | LName    | canSupervise | email                         |
-- +--------+--------+----------+--------------+-------------------------------+
-- |   1000 | Denise | de Vries |            1 | denise.deries@flinders.edu.au |
-- |   1001 | Paul   | Calder   |            1 | paul.calder@flinders.edu.au   |
-- |   1002 | John   | Roddick  |            1 | john.roddick@flinders.edu.au  |
-- |   1003 | Jennie | Brand    |            0 | jennie.brand@flinders.edu.au  |
-- +--------+--------+----------+--------------+-------------------------------+
DROP FUNCTION IF EXISTS CURRENT_RHD_USER $$
CREATE FUNCTION CURRENT_RHD_USER()
RETURNS mediumint
DETERMINISTIC
BEGIN
  RETURN 1000;
--  RETURN 1001;
END $$

DROP FUNCTION IF EXISTS CURRENT_RHD_TIMESTAMP $$
CREATE FUNCTION CURRENT_RHD_TIMESTAMP()
RETURNS char(19)
DETERMINISTIC
BEGIN
  RETURN '2014-05-17 12:00:00';
--  RETURN current_timestamp();
END $$

DROP FUNCTION IF EXISTS CURRENT_RHD_DATE $$
CREATE FUNCTION CURRENT_RHD_DATE()
RETURNS char(10)
DETERMINISTIC
BEGIN
  RETURN DATE(CURRENT_RHD_TIMESTAMP());
END $$

DELIMITER ;
```

4.6.2.2 Views for all staff, including professional staff

```
-- ---------------------------------------------------------------------------
-- For a member of staff, create a view that's shows all the ongoing
-- applications that they're in some way working on.
--
-- In the following, we have hard-coded a user/staff ID of 1000, as we don't
-- have permission in the CSEM MySQL server to create database users. In a real
-- system, we would replace the hard-coded values with the function USER() to
-- get the current database user ID.
SELECT 'Creating views for professional/all staff';

-- All the Applications that the current user is marked as primary supervisor
DROP VIEW IF EXISTS `MyRHDApps_SupervisedPrimaryByMe_Expanded` ;
CREATE VIEW `MyRHDApps_SupervisedPrimaryByMe_Expanded` AS
SELECT 'a) primary supervisor' AS `My role`, App.*
FROM Application_Ongoing_Expanded App
INNER JOIN `Supervise as primary` Super
  ON Super.ApplicationID = App.ApplicationID
WHERE Super.StaffID = CURRENT_RHD_USER() ;

-- ... as associate supervisor
DROP VIEW IF EXISTS `MyRHDApps_SupervisedAssociateByMe_Expanded` ;
CREATE VIEW `MyRHDApps_SupervisedAssociateByMe_Expanded` AS
SELECT 'b) associate supervisor' AS `My role`, App.*
FROM Application_Ongoing_Expanded App
INNER JOIN `Supervise as associate` Super
  ON Super.ApplicationID = App.ApplicationID
WHERE Super.StaffID = CURRENT_RHD_USER() ;

-- All the Applications that the current user has flagged
DROP VIEW IF EXISTS `MyRHDApps_FlaggedByMe_Expanded` ;
CREATE VIEW `MyRHDApps_FlaggedByMe_Expanded` AS
SELECT 'c) flagged by me' AS `My role`, App.*
FROM Application_Ongoing_Expanded App
INNER JOIN `University Staff Member_Application` Flag
  ON Flag.ApplicationID = App.ApplicationID
WHERE Flag.StaffID = CURRENT_RHD_USER()
ORDER BY App.ApplicationID DESC;

-- All the Applications that the current user has most recently modified
DROP VIEW IF EXISTS MyRHDApps_LastModifiedByMe_Expanded ;
CREATE VIEW MyRHDApps_LastModifiedByMe_Expanded AS
SELECT 'd) modified by me' AS `My role`, App.*
FROM Application_Ongoing_Expanded App
WHERE App.LastModifiedByStaffID = CURRENT_RHD_USER()
ORDER BY App.DateLastModified;

-- Composite view of the above
DROP VIEW IF EXISTS `MyRHDApps_Expanded` ;
CREATE VIEW `MyRHDApps_Expanded` AS
SELECT * FROM MyRHDApps_SupervisedPrimaryByMe_Expanded
UNION
SELECT * FROM MyRHDApps_SupervisedAssociateByMe_Expanded
UNION
SELECT * FROM MyRHDApps_FlaggedByMe_Expanded
UNION
SELECT * FROM MyRHDApps_LastModifiedByMe_Expanded;


-- As above, but with irrelevent columns suppressed
DROP VIEW IF EXISTS `MyRHDApps` ;
CREATE VIEW `MyRHDApps` AS
SELECT
  `My role`,
  FName,
  LName,
  Email,
  Status,
  AddressConfirmed,
  DegreeConfirmed,
  VisaStatusConfirmed,
  ProposalConfirmed,
  HasResearchAreas,
  HasPrimarySuper,
  PayMethConfirmed,
  EngProfConfirmed,
  RefereesConfirmed
FROM MyRHDApps_Expanded
GROUP BY ApplicationID
ORDER BY `My role`, LName;
```

4.6.2.3 Views for academic staff

```
-- ----------------------------------------------------------------------------
-- Create views specific to academic staff members
SELECT 'Creating views for academic staff';

-- List all the recent applications in the same research area as the current
-- user has nominated an interest in
DROP VIEW IF EXISTS Recent_Applications_In_My_Research_Area;
CREATE VIEW Recent_Applications_In_My_Research_Area AS
SELECT
  App.DateAdded,
  App.ApplicationID,
  Appt.FName,
  Appt.LName,
  Appt.Email,
  RA.Description AS `Research Area Description`,
  USM.FName AS StaffInResearchAreaFName,
  USM.LName AS StaffInResearchAreaLName
FROM Application App
INNER JOIN Applicant Appt
  ON App.ApplicantID = Appt.ApplicantID
INNER JOIN `Application_Research Area` ARA
  ON App.ApplicationID = ARA.ApplicationID
INNER JOIN `Research Area` AS RA
  ON ARA.FORCode = RA.FORCode
INNER JOIN `University Staff Member_Research Area` USM_RA
  ON ARA.FORCode = USM_RA.FORCode
INNER JOIN `University Staff Member` USM
  ON USM_RA.StaffID = USM.StaffID
WHERE USM_RA.StaffID = CURRENT_RHD_USER()
  AND DATEDIFF(CURRENT_RHD_DATE(), App.DateAdded) >= 7
ORDER BY App.DateAdded;
```

4.6.2.4 Views for RHD administration staff

```
-- ----------------------------------------------------------------------------
-- Views for RHD staff.
SELECT 'Creating views for RHD staff';

-- All ongoing applications and the contact staff
DROP VIEW IF EXISTS Application_Staff_Overview ;
CREATE VIEW Application_Staff_Overview AS
SELECT
  AppExpand.ApplicationID,
  AppExpand.FName,
  AppExpand.LName,
  AppExpand.Email,
  PrimaryUSM.FName AS PrimarySupervisorFName,
  PrimaryUSM.LName AS PrimarySupervisorLName,
  COUNT(AssociateSuper.StaffID) AS `Associate supervisor count`,
  LastModifiedByUSM.FName AS LastModifiedByFName,
  LastModifiedByUSM.LName AS LastModifiedByLName
FROM Application_Ongoing_Expanded AS AppExpand
LEFT JOIN (`Supervise as primary` AS PrimarySuper,
           `University Staff Member` AS PrimaryUSM)
  ON (AppExpand.ApplicationID = PrimarySuper.ApplicationID
      AND PrimarySuper.StaffID = PrimaryUSM.StaffID)
LEFT JOIN (`Supervise as associate` AS AssociateSuper)
  ON (AppExpand.ApplicationID = AssociateSuper.ApplicationID)
LEFT JOIN (`University Staff Member` AS LastModifiedByUSM)
  ON (AppExpand.LastModifiedByStaffID = LastModifiedByUSM.StaffID)
GROUP BY AppExpand.ApplicationID
ORDER BY LName;

-- a utility view for the view below
DROP VIEW IF EXISTS Application_Primary_Supervisor ;
CREATE VIEW Application_Primary_Supervisor AS
SELECT
  App.*,
  Super.StaffID,
  Super.PrimarySupervisor,
  SUM(Super.PrimarySupervisor) AS PriSuperSum
FROM Application_Ongoing_Expanded App
LEFT OUTER JOIN `Supervise as` Super
  ON Super.ApplicationID = App.ApplicationID
GROUP BY App.ApplicationID;

-- List only those ongoing applications that don't yet have a primary supervisor
DROP VIEW IF EXISTS Application_Without_Supervisor_Inner ;
CREATE VIEW Application_Without_Supervisor_Inner AS
SELECT
  AppPri.ApplicationID,
  AppPri.FName,
  AppPri.LName,
  AppPri.Email,
  AppPri.DateAdded,
  COUNT(AssociateSuper.StaffID) AS `Associate supervisor count`
FROM Application_Primary_Supervisor AS AppPri
LEFT JOIN (`Supervise as associate` AS AssociateSuper)
  ON (AppPri.ApplicationID = AssociateSuper.ApplicationID)
WHERE AppPri.PriSuperSum <> 1
  OR AppPri.PriSuperSum IS NULL
GROUP BY AppPri.ApplicationID
ORDER BY DateAdded ;

-- Show those applications that don't yet have a primary supervisor.
-- Where the application has nominated a research area, report which staff
-- member oversees that research area.
DROP VIEW IF EXISTS Application_Without_Supervisor ;
CREATE VIEW Application_Without_Supervisor AS
SELECT
  AppWoSuper.*,
  AppRA.FORCode,
  USM.FName as StaffOverseeingAreaFName,
  USM.LName as StaffOverseeingAreaLName
FROM Application_Without_Supervisor_Inner AppWoSuper
LEFT JOIN (`Application_Research Area` AS AppRA)
  ON (AppWoSuper.ApplicationID = AppRA.ApplicationID)
LEFT JOIN (`Research Area` AS RA)
  ON (AppRA.FORCode = RA.FORCode)
LEFT JOIN (`University Staff Member_Research Area2` AS USM_RA_Oversees)
  ON (RA.FORCode = USM_RA_Oversees.FORCode)
LEFT JOIN (`University Staff Member` AS USM)
  ON (USM_RA_Oversees.StaffID = USM.StaffID)
ORDER BY DateAdded;
```

4.6.3 Create procedure for setting user privileges

```
-- ------------------------------------------------------------------------------
-- security.sql
--
```

```sql
-- Script to assist DB admins to enforce security policies. This needs to be
-- read into the DB with admin privileges.
--
-- -----------------------------------------------------------------------------

DROP PROCEDURE IF EXISTS insert_new_staff ;


DELIMITER $$

CREATE PROCEDURE insert_new_staff(
  IN p_staffID int(10),
  IN p_password varchar(255),
  IN p_fName varchar(50),
  IN p_lName varchar(50),
  IN p_canSupervise int(1),
  IN p_email varchar(100) )
BEGIN

  -- Create a database user.
  -- Have to use dynamic SQL here because of the password.
  SET @create_user_query = CONCAT(
        'CREATE USER "', p_staffID, '" IDENTIFIED BY "', p_password, '" ');
  PREPARE create_user_stmt FROM @create_user_query ;
  EXECUTE create_user_stmt ;
  DEALLOCATE PREPARE create_user_stmt;

  -- add the staff member
  INSERT INTO `University Staff Member` (StaffID, FName, LName, canSupervise,
  Email)
  VALUES (p_staffID, p_fName, p_lName, p_canSupervise, p_email) ;

  -- Set the appropriate permissions:
  --   - no delete permissions
  --   - only select on 'University Staff Member'
  --   - select, update and insert on all other tables
  GRANT SELECT ON TABLE `University Staff Member` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Applicant` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Application` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Application Status` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Application_Research Area` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Award Type` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Correspondence` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Correspondence Method` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Country` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Decision` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Decision Type` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Degree` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Document` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Document Status` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Document Type` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Payment Method` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Publication` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Referee` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Research Area` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Visa` TO p_staffID;
  GRANT SELECT,UPDATE,INSERT ON TABLE `Visa Status` TO p_staffID;

  -- These tables consist (almost) entirely of FKs, so update isn't really
  -- needed, and this makes monitoring changes to these with triggers more
  -- straightforward.
  GRANT SELECT,INSERT,DELETE ON TABLE `Supervise as` TO p_staffID;
  GRANT SELECT,INSERT,DELETE ON TABLE `University Staff Member_Application`
    TO p_staffID;
  GRANT SELECT,INSERT,DELETE ON TABLE `University Staff Member_Research Area`
    TO p_staffID;
  GRANT SELECT,INSERT,DELETE ON TABLE `University Staff Member_Research Area2`
    TO p_staffID;
```

```
END $$

DELIMITER ;
```

4.6.4 Create triggers for tracking changes and enforcing constraints

```
-- ----------------------------------------------------------------------------
-- Create triggers for logging and constraint checking

-- ----------------------------------------------------------------------------
-- Make sure there's only ever one primary supervisor for each application

DELIMITER $$

DROP TRIGGER IF EXISTS checkPrimSuper $$
CREATE TRIGGER checkPrimSuper BEFORE INSERT ON `Supervise as`
FOR EACH ROW
BEGIN
  DECLARE existingCount int default 0;

  SELECT count(*)
  FROM `Supervise as`
  WHERE PrimarySupervisor>0
  AND ApplicationID=NEW.ApplicationID
  INTO existingCount;

  IF existingCount IS NOT NULL THEN
    IF existingCount > 0 THEN
      SET NEW.PrimarySupervisor = 0;
    END IF;
  END IF;
END $$

DELIMITER ;

DELIMITER $$

-- ---------------------------------------------------------------------------
-- Record changes to Research Areas to email these to the change subject later
DROP TRIGGER IF EXISTS USM_RESEARCHAREA_AI $$
CREATE TRIGGER USM_RESEARCHAREA_AI AFTER INSERT ON
`University Staff Member_Research Area`
FOR EACH ROW
BEGIN
  DECLARE v_forCodeDescription varchar(2000) ;
  DECLARE v_comment varchar(1000) ;
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) ;
  DECLARE v_decisionTypeID mediumint(9) ;

  SELECT Description
  FROM `Research Area` RA
  WHERE RA.FORCode = NEW.FORCode
  INTO v_forCodeDescription ;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type = 'change.research_area.addition'
  INTO v_decisionTypeID;
  SET v_comment = CONCAT('A new Field Of Research Code ', NEW.FORCode,
  ' - ''', SUBSTRING(v_forCodeDescription, 1, 800),
  ''' has been associated to your RHD account by ', v_agentFName, ' ',
  v_agentLName);

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, NEW.StaffID, v_decisionTypeID, 1, 0);
END $$
DELIMITER ;

DELIMITER $$
DROP TRIGGER IF EXISTS USM_RESEARCHAREA_AD $$
CREATE TRIGGER USM_RESEARCHAREA_AD AFTER DELETE ON
`University Staff Member_Research Area`
FOR EACH ROW
BEGIN
  DECLARE v_forCodeDescription varchar(2000) ;
  DECLARE v_comment varchar(1000) ;
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) ;
  DECLARE v_decisionTypeID mediumint(9) ;

  SELECT Description
  FROM `Research Area` RA
  WHERE RA.FORCode = OLD.FORCode
  INTO v_forCodeDescription ;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type = 'change.research_area.deletion'
  INTO v_decisionTypeID;

  SET v_comment = CONCAT('An existing Field Of Research Code ', OLD.FORCode,
  ' - ''', SUBSTRING(v_forCodeDescription, 1, 800),
  ''' has been removed from association with your RHD account by ',
  v_agentFName, ' ', v_agentLName);

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, OLD.StaffID, v_decisionTypeID, 1, 0);
END $$
DELIMITER ;

DELIMITER $$
-- -------------------------------------------------------------------------------
-- Record changes to Research Areas to email these to the change subject later
DROP TRIGGER IF EXISTS USM_RESEARCHAREA2_AI $$
CREATE TRIGGER USM_RESEARCHAREA2_AI AFTER INSERT ON
`University Staff Member_Research Area2`
FOR EACH ROW
BEGIN
  DECLARE v_forCodeDescription varchar(2000) ;
  DECLARE v_comment varchar(1000) ;
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) ;
  DECLARE v_decisionTypeID mediumint(9) ;

  SELECT Description
  FROM `Research Area` RA
  WHERE RA.FORCode = NEW.FORCode
  INTO v_forCodeDescription ;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type LIKE 'change.research_area_oversees.addition'
  INTO v_decisionTypeID;

  SET v_comment = CONCAT(
  'You have been registered as overseeing Field Of Research area ', NEW.FORCode,
  ' - ''', SUBSTRING(v_forCodeDescription, 1, 800),
  ''', this change made by ', v_agentFName, ' ',
  v_agentLName);

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, NEW.StaffID, v_decisionTypeID, 1, 0);
END $$
DELIMITER ;

DELIMITER $$
DROP TRIGGER IF EXISTS USM_RESEARCHAREA2_AD $$
CREATE TRIGGER USM_RESEARCHAREA2_AD AFTER DELETE ON
`University Staff Member_Research Area2`
FOR EACH ROW
BEGIN
  DECLARE v_forCodeDescription varchar(2000) ;
  DECLARE v_comment varchar(1000) ;
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) ;
  DECLARE v_decisionTypeID mediumint(9) ;

  SELECT Description
  FROM `Research Area` RA
  WHERE RA.FORCode = OLD.FORCode
  INTO v_forCodeDescription ;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type = 'change.research_area_oversees.deletion'
  INTO v_decisionTypeID;

  SET v_comment = CONCAT(
  'You have been deregistered as overseeing Field Of Research area ',
  OLD.FORCode, ' - ''', SUBSTRING(v_forCodeDescription, 1, 800),
  ''', this change made by ', v_agentFName, ' ', v_agentLName);

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, OLD.StaffID, v_decisionTypeID, 1, 0);
END $$

DELIMITER ;

-- ---------------------------------------------------------------------------
-- Add a triggers to record changes to supervisors
DELIMITER $$
DROP TRIGGER IF EXISTS SUPERVISE_AS_AI $$
CREATE TRIGGER SUPERVISE_AS_AI AFTER INSERT ON
`Supervise as`
FOR EACH ROW
BEGIN
  DECLARE v_apptFName varchar(50) ;
  DECLARE v_apptLName varchar(50) ;
  DECLARE v_apptEmail varchar(100) ;
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) ;
  DECLARE v_decisionTypeID mediumint(9) ;
  DECLARE v_supervisionRole varchar(20) DEFAULT 'an associate';
  DECLARE v_comment varchar(1000) ;

  SELECT Appt.FName, Appt.LName, Appt.Email
  FROM Application App
  INNER JOIN Applicant Appt ON (App.ApplicantID = Appt.ApplicantID)
  WHERE App.ApplicationID = NEW.ApplicationID
  INTO v_apptFName, v_apptLName, v_apptEmail ;

  IF v_apptLName IS NULL THEN
    SET v_apptLName = ' (no lastname registered)';
  END IF;
  IF v_apptEmail IS NULL THEN
    SET v_apptEmail = ' (no email registered)';
  END IF;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type LIKE 'change.supervisor.addition'
  INTO v_decisionTypeID;

  IF NEW.PrimarySupervisor = 1 THEN
    SET v_supervisionRole = 'the primary';
  END IF;

  SET v_comment = CONCAT(
  'You have been registered as ', v_supervisionRole,
  ' supervisor for an RHD application from ', v_apptFName, ' ', v_apptLName,
  ' (application ID ', NEW.ApplicationID, '). This change made by ',
  v_agentFName, ' ', v_agentLName, '.');

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, NEW.StaffID, v_decisionTypeID, 1, 0);
END $$
DELIMITER ;

DELIMITER $$
DROP TRIGGER IF EXISTS SUPERVISE_AS_AD $$
CREATE TRIGGER SUPERVISE_AS_AD AFTER DELETE ON
`Supervise as`
FOR EACH ROW
BEGIN
  DECLARE v_apptFName varchar(50) ;
  DECLARE v_apptLName varchar(50) ;
  DECLARE v_apptEmail varchar(100) ;
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) ;
  DECLARE v_decisionTypeID mediumint(9) ;
  DECLARE v_supervisionRole varchar(20) DEFAULT 'an associate';
  DECLARE v_comment varchar(1000) ;

  SELECT Appt.FName, Appt.LName, Appt.Email
  FROM Application App
  INNER JOIN Applicant Appt ON (App.ApplicantID = Appt.ApplicantID)
  WHERE App.ApplicationID = OLD.ApplicationID
  INTO v_apptFName, v_apptLName, v_apptEmail ;

  IF v_apptLName IS NULL THEN
    SET v_apptLName = ' (no lastname registered)';
  END IF;
  IF v_apptEmail IS NULL THEN
    SET v_apptEmail = ' (no email registered)';
  END IF;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type LIKE 'change.supervisor.addition'
  INTO v_decisionTypeID;

  IF OLD.PrimarySupervisor = 1 THEN
    SET v_supervisionRole = 'the primary';
  END IF;

  SET v_comment = CONCAT(
  'You have been deregistered as ', v_supervisionRole,
  ' supervisor for an RHD application from ', v_apptFName, ' ', v_apptLName,
  ' (application ID ', OLD.ApplicationID, '). This change made by ',
  v_agentFName, ' ', v_agentLName, '.');

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, OLD.StaffID, v_decisionTypeID, 1, 0);
END $$
DELIMITER ;


-- ---------------------------------------------------------------------------------
-- Record changes to 'flagging' an application.
-- Adding a flag
DELIMITER $$
DROP TRIGGER IF EXISTS USM_APPLICATION_AI $$
CREATE TRIGGER USM_APPLICATION_AI AFTER INSERT ON
`University Staff Member_Application`
FOR EACH ROW
BEGIN
  DECLARE v_apptFName varchar(50) ;
  DECLARE v_apptLName varchar(50) ;
  DECLARE v_apptEmail varchar(100) ;
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) ;
  DECLARE v_decisionTypeID mediumint(9) ;
  DECLARE v_emailUpdates varchar(100) DEFAULT ' not';
  DECLARE v_comment varchar(1000) ;

  SELECT Appt.FName, Appt.LName, Appt.Email
  FROM Application App
  INNER JOIN Applicant Appt ON (App.ApplicantID = Appt.ApplicantID)
  WHERE App.ApplicationID = NEW.ApplicationID
  INTO v_apptFName, v_apptLName, v_apptEmail ;

  IF v_apptLName IS NULL THEN
    SET v_apptLName = ' (no lastname registered)';
  END IF;
  IF v_apptEmail IS NULL THEN
    SET v_apptEmail = ' (no email registered)';
  END IF;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type LIKE 'change.flag.addition'
  INTO v_decisionTypeID;

  IF NEW.ReceiveEmailUpdates = 1 THEN
    SET v_emailUpdates = '';
  END IF;

  SET v_comment = CONCAT(
  'An flag for you on an RHD application from ',
  v_apptFName, ' ', v_apptLName,
  ' (application ID ', NEW.ApplicationID, ') has been added. ',
  'You are', v_emailUpdates,
  ' registered for email updates to this application. ',
  'This change made by ', v_agentFName, ' ', v_agentLName, '.');

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, NEW.StaffID, v_decisionTypeID, 1, 0);
END $$
DELIMITER ;

-- --------------------------------------------------------------------------------
-- Record changes to 'flagging' an application.
-- Removing a flag.
DELIMITER $$
DROP TRIGGER IF EXISTS USM_APPLICATION_AD $$
CREATE TRIGGER USM_APPLICATION_AD AFTER DELETE ON
`University Staff Member_Application`
FOR EACH ROW
BEGIN
  DECLARE v_apptFName varchar(50) ;
  DECLARE v_apptLName varchar(50) DEFAULT ' (no lastname registered)';
  DECLARE v_apptEmail varchar(100) DEFAULT ' (no email registered)';
  DECLARE v_agentFName varchar(50) ;
  DECLARE v_agentLName varchar(50) DEFAULT ' (no lastname registered)';
  DECLARE v_decisionTypeID mediumint(9) ;
  DECLARE v_comment varchar(1000) ;

  SELECT Appt.FName, Appt.LName, Appt.Email
  FROM Application App
  INNER JOIN Applicant Appt ON (App.ApplicantID = Appt.ApplicantID)
  WHERE App.ApplicationID = OLD.ApplicationID
  INTO v_apptFName, v_apptLName, v_apptEmail ;

  IF v_apptLName IS NULL THEN
    SET v_apptLName = ' (no lastname registered)';
  END IF;
  IF v_apptEmail IS NULL THEN
    SET v_apptEmail = ' (no email registered)';
  END IF;

  SELECT FName, LName
  FROM `University Staff Member` AS USM
  WHERE USM.StaffID = CURRENT_RHD_USER()
  INTO v_agentFName, v_agentLName;

  SELECT DecisionTypeID
  FROM `Decision Type` AS DT
  WHERE DT.type LIKE 'change.flag.addition'
  INTO v_decisionTypeID;

  SET v_comment = CONCAT(
  'Your flagged interest in an RHD application from ',
  v_apptFName, ' ', v_apptLName,
  ' (application ID ', OLD.ApplicationID, ') has been removed. ',
  'This change made by ', v_agentFName, ' ', v_agentLName, '.');

  INSERT INTO Decision(Date, Comment, StaffID, DecisionTypeID, Reportable, Sent)
  VALUES (CURRENT_DATE(), v_comment, OLD.StaffID, v_decisionTypeID, 1, 0);
END $$
DELIMITER ;

-- ------------------------------------------------------------------------------
--  Applicant emails are automatically inserted into new Applications
DELIMITER $$
DROP TRIGGER IF EXISTS APPLICATION_EMAIL_INSERT $$
CREATE TRIGGER APPLICATION_EMAIL_INSERT BEFORE INSERT ON `Application`
FOR EACH ROW
BEGIN
SET NEW.ApplicantEmail=(SELECT Email FROM Applicant WHERE
ApplicantID=New.ApplicantID);
END $$
DELIMITER ;

-- ------------------------------------------------------------------------------
--  UPDATE of Applicant emails are automatically updated for all Applications
DELIMITER $$
DROP TRIGGER IF EXISTS APPLICANT_EMAIL_UPDATE $$
CREATE TRIGGER APPLICANT_EMAIL_UPDATE AFTER UPDATE ON `Applicant`
FOR EACH ROW
BEGIN
UPDATE `Application` SET ApplicantEmail=NEW.Email WHERE
Application.ApplicantID=NEW.ApplicantID ;
END $$
DELIMITER ;
```

4.7 Create SQL scripts to populate all tables with data

4.7.1 Populate Country relation

```
-- ---------------------------------------------------------------------------
-- Country
SELECT "Loading populate_country.sql" ;

INSERT INTO Country (CountryISOCode, Name)
VALUES
('AD', 'Andorra'),
('AE', 'United Arab Emirates'),
('AF', 'Afghanistan'),
('AG', 'Antigua and Barbuda'),
('AI', 'Anguilla'),
('AL', 'Albania'),
('AM', 'Armenia'),
('AO', 'Angola'),
('AQ', 'Antarctica'),
('AR', 'Argentina'),
('AS', 'American Samoa'),
('AT', 'Austria'),
('AU', 'Australia'),
('AW', 'Aruba'),
('AX', 'Åland Islands'),
('AZ', 'Azerbaijan'),
('BA', 'Bosnia and Herzegovina'),
('BB', 'Barbados'),
('BD', 'Bangladesh'),
('BE', 'Belgium'),
('BF', 'Burkina Faso'),
('BG', 'Bulgaria'),
('BH', 'Bahrain'),
('BI', 'Burundi'),
('BJ', 'Benin'),
('BL', 'Saint Barthélemy'),
('BM', 'Bermuda'),
('BN', 'Brunei Darussalam'),
('BO', 'Bolivia, Plurinational State of'),
('BQ', 'Bonaire, Sint Eustatius and Saba'),
('BR', 'Brazil'),
('BS', 'Bahamas'),
('BT', 'Bhutan'),
('BV', 'Bouvet Island'),
('BW', 'Botswana'),
('BY', 'Belarus'),
('BZ', 'Belize'),
('CA', 'Canada'),
('CC', 'Cocos (Keeling) Islands'),
('CD', 'Congo, the Democratic Republic of the'),
('CF', 'Central African Republic'),
('CG', 'Congo'),
('CH', 'Switzerland'),
('CI', 'Côte d''Ivoire'),
('CK', 'Cook Islands'),
('CL', 'Chile'),
('CM', 'Cameroon'),
('CN', 'China'),
('CO', 'Colombia'),
('CR', 'Costa Rica'),
('CU', 'Cuba'),
('CV', 'Cabo Verde'),
('CW', 'Curaçao'),
('CX', 'Christmas Island'),
('CY', 'Cyprus'),
('CZ', 'Czech Republic'),
('DE', 'Germany'),
('DJ', 'Djibouti'),
('DK', 'Denmark'),
('DM', 'Dominica'),
('DO', 'Dominican Republic'),
('DZ', 'Algeria'),
('EC', 'Ecuador'),
('EE', 'Estonia'),
('EG', 'Egypt'),
('EH', 'Western Sahara'),
('ER', 'Eritrea'),
('ES', 'Spain'),
('ET', 'Ethiopia'),
('FI', 'Finland'),
('FJ', 'Fiji'),
('FK', 'Falkland Islands (Malvinas)'),
('FM', 'Micronesia, Federated States of'),
('FO', 'Faroe Islands'),
('FR', 'France'),
('GA', 'Gabon'),
('GB', 'United Kingdom'),
('GD', 'Grenada'),
('GE', 'Georgia'),
('GF', 'French Guiana'),
('GG', 'Guernsey'),
('GH', 'Ghana'),
('GI', 'Gibraltar'),
('GL', 'Greenland'),
('GM', 'Gambia'),
('GN', 'Guinea'),
('GP', 'Guadeloupe'),
('GQ', 'Equatorial Guinea'),
('GR', 'Greece'),
('GS', 'South Georgia and the South Sandwich Islands'),
('GT', 'Guatemala'),
('GU', 'Guam'),
('GW', 'Guinea-Bissau'),
('GY', 'Guyana'),
('HK', 'Hong Kong'),
('HM', 'Heard Island and McDonald Islands'),
('HN', 'Honduras'),
('HR', 'Croatia'),
('HT', 'Haiti'),
('HU', 'Hungary'),
('ID', 'Indonesia'),
('IE', 'Ireland'),
('IL', 'Israel'),
('IM', 'Isle of Man'),
('IN', 'India'),
('IO', 'British Indian Ocean Territory'),
('IQ', 'Iraq'),
('IR', 'Iran, Islamic Republic of'),
('IS', 'Iceland'),
('IT', 'Italy'),
('JE', 'Jersey'),
('JM', 'Jamaica'),
('JO', 'Jordan'),
('JP', 'Japan'),
('KE', 'Kenya'),
('KG', 'Kyrgyzstan'),
('KH', 'Cambodia'),
('KI', 'Kiribati'),
('KM', 'Comoros'),
('KN', 'Saint Kitts and Nevis'),
('KP', 'Korea, Democratic People''s Republic of'),
('KR', 'Korea, Republic of'),
('KW', 'Kuwait'),
('KY', 'Cayman Islands'),
('KZ', 'Kazakhstan'),
('LA', 'Lao People''s Democratic Republic'),
('LB', 'Lebanon'),
('LC', 'Saint Lucia'),
('LI', 'Liechtenstein'),
('LK', 'Sri Lanka'),
('LR', 'Liberia'),
('LS', 'Lesotho'),
('LT', 'Lithuania'),
('LU', 'Luxembourg'),
('LV', 'Latvia'),
('LY', 'Libya'),
('MA', 'Morocco'),
('MC', 'Monaco'),
('MD', 'Moldova, Republic of'),
('ME', 'Montenegro'),
('MF', 'Saint Martin (French part)'),
('MG', 'Madagascar'),
('MH', 'Marshall Islands'),
('MK', 'Macedonia, the former Yugoslav Republic of'),
('ML', 'Mali'),
('MM', 'Myanmar'),
('MN', 'Mongolia'),
('MO', 'Macao'),
('MP', 'Northern Mariana Islands'),
('MQ', 'Martinique'),
('MR', 'Mauritania'),
('MS', 'Montserrat'),
('MT', 'Malta'),
('MU', 'Mauritius'),
('MV', 'Maldives'),
('MW', 'Malawi'),
('MX', 'Mexico'),
('MY', 'Malaysia'),
('MZ', 'Mozambique'),
('NA', 'Namibia'),
('NC', 'New Caledonia'),
('NE', 'Niger'),
('NF', 'Norfolk Island'),
('NG', 'Nigeria'),
('NI', 'Nicaragua'),
('NL', 'Netherlands'),
('NO', 'Norway'),
('NP', 'Nepal'),
('NR', 'Nauru'),
('NU', 'Niue'),
('NZ', 'New Zealand'),
('OM', 'Oman'),
('PA', 'Panama'),
('PE', 'Peru'),
('PF', 'French Polynesia'),
('PG', 'Papua New Guinea'),
('PH', 'Philippines'),
('PK', 'Pakistan'),
('PL', 'Poland'),
('PM', 'Saint Pierre and Miquelon'),
('PN', 'Pitcairn'),
('PR', 'Puerto Rico'),
('PS', 'Palestine, State of'),
('PT', 'Portugal'),
('PW', 'Palau'),
('PY', 'Paraguay'),
('QA', 'Qatar'),
('RE', 'Réunion'),
('RO', 'Romania'),
('RS', 'Serbia'),
('RU', 'Russian Federation'),
('RW', 'Rwanda'),
('SA', 'Saudi Arabia'),
('SB', 'Solomon Islands'),
('SC', 'Seychelles'),
('SD', 'Sudan'),
('SE', 'Sweden'),
('SG', 'Singapore'),
('SH', 'Saint Helena, Ascension and Tristan da Cunha'),
('SI', 'Slovenia'),
('SJ', 'Svalbard and Jan Mayen'),
('SK', 'Slovakia'),
('SL', 'Sierra Leone'),
('SM', 'San Marino'),
('SN', 'Senegal'),
('SO', 'Somalia'),
('SR', 'Suriname'),
('SS', 'South Sudan'),
('ST', 'Sao Tome and Principe'),
('SV', 'El Salvador'),
('SX', 'Sint Maarten (Dutch part)'),
('SY', 'Syrian Arab Republic'),
('SZ', 'Swaziland'),
('TC', 'Turks and Caicos Islands'),
('TD', 'Chad'),
('TF', 'French Southern Territories'),
('TG', 'Togo'),
('TH', 'Thailand'),
('TJ', 'Tajikistan'),
('TK', 'Tokelau'),
('TL', 'Timor-Leste'),
('TM', 'Turkmenistan'),
('TN', 'Tunisia'),
('TO', 'Tonga'),
('TR', 'Turkey'),
('TT', 'Trinidad and Tobago'),
('TV', 'Tuvalu'),
('TW', 'Taiwan, Province of China'),
('TZ', 'Tanzania, United Republic of'),
('UA', 'Ukraine'),
('UG', 'Uganda'),
('UM', 'United States Minor Outlying Islands'),
('US', 'United States'),
('UY', 'Uruguay'),
('UZ', 'Uzbekistan'),
('VA', 'Holy See (Vatican City State)'),
('VC', 'Saint Vincent and the Grenadines'),
('VE', 'Venezuela, Bolivarian Republic of'),
('VG', 'Virgin Islands, British'),
('VI', 'Virgin Islands, U.S.'),
('VN', 'Viet Nam'),
('VU', 'Vanuatu'),
('WF', 'Wallis and Futuna'),
('WS', 'Samoa'),
('YE', 'Yemen'),
('YT', 'Mayotte'),
('ZA', 'South Africa'),
('ZM', 'Zambia'),
('ZW', 'Zimbabwe');
```

4.7.2 Populate lookup relations

```
SELECT "Loading populate_lookups.sql" ;

-- ---------------------------------------------------------------------------
-- ApplicationStatus
-- [10000..10499] reserved for all kinds of ongoing statuses
-- [10500..10999] reserved for all kinds of completed statuses
INSERT INTO `Application Status` (ApplicationStatusID, Status, Description)
```

```
VALUES (10000, 'ongoing',
'Application/information gathering is currently ongoing' ) ;

INSERT INTO `Application Status` (ApplicationStatusID, Status, Description)
VALUES (10500, 'complete.accepted',
'Application accepted. Elevated to RHD office.' ) ;

INSERT INTO `Application Status` (ApplicationStatusID, Status, Description)
VALUES (10501, 'complete.declined',
'Application declined. School chooses not to pursue.' ) ;

INSERT INTO `Application Status` (ApplicationStatusID, Status, Description)
VALUES (10502, 'complete.withdrawn', 'Application withdrawn by applicant.' ) ;

INSERT INTO `Application Status` (ApplicationStatusID, Status, Description)
VALUES (10503, 'complete.lapsed', 'No activity for a significant period.' ) ;


-- ------------------------------------------------------------------------------
-- DocumentType
INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20000, 'application', 'A completed RHD application form.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20001, 'cv', 'CV') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20002, 'resume', 'resume') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20003, 'faculty_assessment_memo', 'Faculty assessment memo') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20004, 'certificate', 'A certificate of a previous degree.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20005, 'transcript', 'An academic transcript of a previous degree.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20006, 'thesis', 'An previous degree major work.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20007, 'proposal', 'An application proposal.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20008, 'reference',
'A character/academic reference of the applicant.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20009, 'publication',
'A scientific publication authored by the applicant.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20010, 'financial',
'A document relating to how the RHD place is to be funded.') ;

INSERT INTO `Document Type` (DocTypeID, Type, Description)
VALUES (20011, 'general', 'A type of document other than decribed above.') ;


-- ------------------------------------------------------------------------------
-- DocumentStatus
INSERT INTO `Document Status` (DocStatusID, Status, Description)
VALUES (30000, 'original.english', 'Original document in English');

INSERT INTO `Document Status` (DocStatusID, Status, Description)
VALUES (30001, 'original.lote', 'Original document in a language other than
English.');
```

```
INSERT INTO `Document Status` (DocStatusID, Status, Description)
VALUES (30002, 'translation', 'Official translation into English of original
document');

INSERT INTO `Document Status` (DocStatusID, Status, Description)
VALUES (30003, 'summary', 'Not an original source document.');


-- ---------------------------------------------------------------------------
-- AwardType
INSERT INTO `Award Type` (AwardID, Type, Description)
VALUES (40000, 'PhD', 'PhD in any field in the school');

INSERT INTO `Award Type` (AwardID, Type, Description)
VALUES (40001, 'PhD (Comp Sc)', 'PhD in Computer Science');

INSERT INTO `Award Type` (AwardID, Type, Description)
VALUES (40002, 'masters', 'Masters by research in any field in the school');

INSERT INTO `Award Type` (AwardID, Type, Description)
VALUES (40003, 'MIT', 'Masters IT');


-- ---------------------------------------------------------------------------
-- ResearchArea
INSERT INTO `Research Area` (FORCode, Description, ResearchArea, GeneralArea)
VALUES (100503, 'computer communication networks', '', '');

INSERT INTO `Research Area` (FORCode, Description, ResearchArea, GeneralArea)
VALUES (080199, 'artificial intelligence and image processing not elsewhere
classified', '', '');

INSERT INTO `Research Area` (FORCode, Description, ResearchArea, GeneralArea)
VALUES (080399, 'computer software not elsewhere classified', '', '');

INSERT INTO `Research Area` (FORCode, Description, ResearchArea, GeneralArea)
VALUES (080499, 'data format not elsewhere classified', '', '');

INSERT INTO `Research Area` (FORCode, Description, ResearchArea, GeneralArea)
VALUES (080699, 'information systems not elsewhere classified', '', '');

INSERT INTO `Research Area` (FORCode, Description, ResearchArea, GeneralArea)
VALUES (089999, 'information and computing sciences not elsewhere classified', '',
'');


-- ---------------------------------------------------------------------------
-- DecisionTypes
INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50000, 'GPA too low');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50001, 'TOEFL/IELTS score too low');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50002, 'Hand over');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50003, 'RFI');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50500, 'change.research_area.addition');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50501, 'change.research_area.deletion');

INSERT INTO `Decision Type` (DecisionTypeID, type)
```

```
VALUES (50502, 'change.research_area_oversees.addition');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50503, 'change.research_area_oversees.deletion');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50504, 'change.supervisor.addition');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50505, 'change.supervisor.deletion');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50506, 'change.flag.addition');

INSERT INTO `Decision Type` (DecisionTypeID, type)
VALUES (50507, 'change.flag.deletion');


-- ----------------------------------------------------------------------------
-- CorrespondenceMethod
INSERT INTO `Correspondence Method` (CorrMethodID, Method)
VALUES (60000, 'email');

INSERT INTO `Correspondence Method` (CorrMethodID, Method)
VALUES (60001, 'telephone');

INSERT INTO `Correspondence Method` (CorrMethodID, Method)
VALUES (60002, 'carrier pigeon');

INSERT INTO `Correspondence Method` (CorrMethodID, Method)
VALUES (69999, 'other');


-- ----------------------------------------------------------------------------
-- Payment Method
INSERT INTO `Payment Method` (PayMethodID, Method)
VALUES (70000, 'Savings'), (70001, 'scholarship'), (70002, 'HECS-HELP'),
(70003, 'Research Training Scheme (RTS)');


-- ----------------------------------------------------------------------------
-- Visa Status
INSERT INTO `Visa Status` (VisaStatusID, Status, description)
VALUES (80000,'not required', 'the applicant is an australian/NZ resident and does
not require a visa'),
(80001,'unknown', 'the students visa status is not currently known'),
(80002, 'approved-nodoc', 'Stated the visa has been approved but has not supplied a
document to that effect'),
(80003, 'approved-doc', 'The visa has been approved and has supplied a document
that proves it'),
(80004, 'submitted', 'an application has been submitted and is currently being
processed');
```

4.7.3 Populate staff information

```
-- ----------------------------------------------------------------------------
-- UNIVERSITY STAFF MEMBERS
-- ----------------------------------------------------------------------------
SELECT "Loading populate_staff.sql" ;

-- Denise de Vries
INSERT INTO `University Staff Member` (StaffID, LName, FName, canSupervise,
email)
VALUES (1000, 'de Vries', 'Denise', 1, 'denise.deries@flinders.edu.au');
```

```sql
-- FOR 'computer software not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1000, 080399);

-- FOR 'data format not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1000, 080499);

-- FOR 'information systems not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1000, 080699);

-- FOR 'information and computing sciences not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1000, 089999);

-- Denise oversees data format
INSERT INTO `University Staff Member_Research Area2` (StaffID, FORCode)
VALUES (1000, 080499);

-- Denise oversees info systems
INSERT INTO `University Staff Member_Research Area2` (StaffID, FORCode)
VALUES (1000, 89999);


-- ----------------------------------------------------------------------------
-- Paul Calder
INSERT INTO `University Staff Member` (StaffID, LName, FName, canSupervise,
email)
VALUES (1001, 'Calder', 'Paul', 1, 'paul.calder@flinders.edu.au');

-- FOR 'artificial intelligence and image processing not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1001, 080199);

-- FOR 'computer software not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1001, 080399);

-- FOR 'computer communication networks'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1001, 100503);

-- Paul oversees computer software
INSERT INTO `University Staff Member_Research Area2` (StaffID, FORCode)
VALUES (1001, 080399);

-- Paul oversees 'computer communication networks'
INSERT INTO `University Staff Member_Research Area2` (StaffID, FORCode)
VALUES (1001, 100503);

-- ----------------------------------------------------------------------------
-- John Roddick
INSERT INTO `University Staff Member` (StaffID, LName, FName, canSupervise,
email)
VALUES (1002, 'Roddick', 'John', 1, 'john.roddick@flinders.edu.au');

-- FOR 'artificial intelligence and image processing not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1002, 080199);

-- FOR 'information systems not elsewhere classified'
INSERT INTO `University Staff Member_Research Area` (StaffID, FORCode)
VALUES (1002, 080699);

-- John Roddick oversees AI
INSERT INTO `University Staff Member_Research Area2` (StaffID, FORCode)
VALUES (1001, 80199);
```

```
-- ---------------------------------------------------------------------------
-- Jennie Brand
INSERT INTO `University Staff Member` (StaffID, LName, FName, canSupervise,
email)
VALUES (1003, 'Brand', 'Jennie', 0, 'jennie.brand@flinders.edu.au');
```

4.7.4 Populate applicant and applicant information

```
SELECT "Loading populate_apps.sql" ;

-- IDs
--   Applicant 1-9-
--   Application x11 - x19
--   Degree      x21 - x29
--   Document    x31 - x39
--   Correspondence x41 - x49

-- ---------------------------------------------------------------------------
-- Reset tables and import other info
DELETE FROM `Application_Research Area`;
DELETE FROM Decision;
DELETE FROM Degree;
DELETE FROM Document;
DELETE FROM Application;
DELETE FROM Applicant;
DELETE FROM Correspondence;

-- ---------------------------------------------------------------------------
-- Application for PhD studies.txt
-- 01
SELECT "Populating Applicant 1 info" ;

INSERT INTO Applicant (ApplicantID, FName, LName, Email, Mobile, StreetAddress,
City, Postcode, AddressCountryISOCode, DateAdded, LastModifiedByStaffID)
VALUES (01, 'Shirin', 'Ebadi', 'shirin.ebadi@keb.com.de',
'+49 (176) 6488 9999', 'Zwillingstr 99, App 099', 'Munich', 80807, 'DE',
'2014-05-01', 1001);

INSERT INTO Degree (DegID, ApplicantID, Name, Type, YearCompleted,
InstitutionName, InstitutionCountryISOCode)
VALUES (121, 01, 'Electrical Engineering with expertise in Control
Theory and Reat-Time Applications', 'bachelor', '2005-12-31',
'University of Tabriz', 'IR') ;

INSERT INTO Degree (DegID, ApplicantID, Name, Type, YearCompleted,
InstitutionName, InstitutionCountryISOCode)
VALUES (122, 01, 'Electrical Engineering with expertise in Control
Theory and Reat-Time Applications', 'masters', '2008-12-31',
'University of Tabriz', 'IR') ;

INSERT INTO Application (ApplicationID, ApplicantID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUES (111, 01, 40000, '2014-05-01', '2014-05-01', '2014-05-01', 1002, 10000) ;

INSERT INTO Document (DocID, UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES (131, '/mnt/data/rhd/01/CV/001.pdf', 01, 20001, 30000);

-- Denise to supervise this
INSERT INTO `Supervise as` (PrimarySupervisor, ApplicationID, StaffID)
VALUES (1, 111, 1000);


-- ---------------------------------------------------------------------------
-- FW Your kind supervision for my intended PhD.txt
-- 2
SELECT "Populating Applicant 2 info" ;

INSERT INTO Applicant (ApplicantID, FName, LName, Email, Mobile, Phone,
DateAdded, LastModifiedByStaffID)
VALUES (2, 'Mohammad', 'Almalki', 'don.memo@hotmail.com', '+966 565907070',
'+966 12 527000000 ext 4951', '2014-05-02', 1003);

INSERT INTO Application (ApplicantID, ApplicationID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUES (2, 211, 40000, '2014-05-02', '2014-05-02', '2014-05-02', 1003, 10000) ;

INSERT INTO Degree (DegID, ApplicantID, Name, Type, YearCompleted,
InstitutionName, InstitutionCountryISOCode)
VALUES (221, 2, 'IT', 'master', '2010-12-31',
'University of Technology Sydney', 'AU') ;

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/02/CV/001.pdf',
02, 20001, 30000) ;

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/02/transcript/001.pdf',
02, 20005, 'transcript of masters course', 30000) ;

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/02/certificate/001.pdf',
02, 20004, 'bachelors certificate', 30000) ;

INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
DocStatusID)
VALUES ('/mnt/data/rhd/02/proposal/001.pdf',
02, 211, 20007, 30000) ;

-- Denise to supervise this (associate supervision)
INSERT INTO `Supervise as` (PrimarySupervisor, ApplicationID, StaffID)
VALUES (0, 211, 1000);

-- ---------------------------------------------------------------------------
-- Fwd Flinders Application - PhD (Comp Sc) - Sem 2 2015.txt
-- 3
SELECT "Populating Applicant 3 info" ;

INSERT INTO Applicant (ApplicantID, FName, LName, StudentID, DateAdded,
LastModifiedByStaffID)
VALUES (03, 'Azzam', 'Alwash', '1234567', '2014-05-03', 1003) ;

INSERT INTO Application (ApplicationID, ApplicantID, awardID, ProposalSummary,
ProposedStartDate, DateAdded, DateLastChecked, DateLastModified,
LastModifiedByStaffID, applicationStatusID)
VALUES (311, 03, 40001,
'Genetic algorithms for Arabic character recognition', '2015-07-01',
'2014-05-03', '2014-05-03', '2014-05-03', 1001, 10000) ;

INSERT INTO Degree (ApplicantID, Name, Type, GPA, InstitutionCountryISOCode)
VALUES (03, 'IS and CS', 'bachelors', 4.27, 'IQ') ;

INSERT INTO Degree (ApplicantID, Name, Type, GPA, InstitutionCountryISOCode)
VALUES (03, 'IT', 'masters', 6.79, 'MY') ;

INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
DocStatusID)
VALUES ('/mnt/data/rhd/03/Application/0001.pdf', 03, 311, 20000, 30000);

-- Denise to supervise this
INSERT INTO `Supervise as` (PrimarySupervisor, ApplicationID, StaffID)
```

```
VALUES (1, 311, 1000);

-- Paul Calder to supervise this (associate)
INSERT INTO `Supervise as` (PrimarySupervisor, ApplicationID, StaffID)
VALUES (0, 311, 1001);

-- John Roddick to supervise this (associate)
INSERT INTO `Supervise as` (PrimarySupervisor, ApplicationID, StaffID)
VALUES (0, 311, 1002);

-- ---------------------------------------------------------------------------
-- Fwd Flinders Application - PhD (Computer Science) Sem 2 2014 .txt
-- 4
SELECT "Populating Applicant 4 info" ;

INSERT INTO Applicant (ApplicantID, FName, LName, StudentID, DateAdded,
LastModifiedByStaffID)
VALUES (04, 'Mustafa', 'Al Lami', 2130106, '2014-05-04', 1000);

INSERT INTO Application (ApplicationID, ApplicantID, awardID, ProposalSummary,
DateAdded, DateLastChecked, DateLastModified, LastModifiedByStaffID,
applicationStatusID)
VALUES (411, 04, 40000, 'Cloud computing for large scale data analysis',
'2014-05-04', '2014-05-04', '2014-05-04', 1003, 10000) ;

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/04/faculty_assessment_memo/0001.pdf',
04, 20003, 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/04/general/0001.pdf',
04, 20011, 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/04/general/0002.pdf', 04, 20011, 30000);

-- Paul Colder asks for more info about masters
INSERT INTO Decision ( DecID, ApplicationID, Date, DecisionTypeID, Comment,
StaffID)
VALUES (1, 411, '2014.05.15', 50003,
'Have asked for more info about Masters project', 1001 );

INSERT INTO Correspondence (CorrID, `Date`, CorrMethodID, Summary,
ApplicationID, StaffID)
VALUES (441, '2014-05-04', 60000, 'Initial inquiry from applicant', 411, 1001);

INSERT INTO Correspondence (CorrID, `Date`, CorrMethodID, Summary,
ApplicationID, StaffID)
VALUES (442, '2014-05-04', 60000, 'Asked applicant for more information', 411,
1001);


-- ---------------------------------------------------------------------------
-- Fwd Flinders Application.txt
-- 5
SELECT "Populating Applicant 5 info" ;

INSERT INTO Applicant (ApplicantID, FName, StudentID, DateAdded,
LastModifiedByStaffID)
VALUES (05, 'Ena', '2345678', '2014-05-05', 1001) ;

INSERT INTO Application (ApplicationID, ApplicantID, ProposedStartDate,
DateAdded, DateLastChecked, DateLastModified, LastModifiedByStaffID,
applicationStatusID)
VALUES (511, 05, '2014-07-01', '2014-05-05', '2014-05-05', '2014-05-05', 1001,
10000) ;

INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
DocStatusID)
VALUES ('/mnt/data/rhd/05/proposal/0001.pdf', 05, 511, 20007, 30000);

INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
DocStatusID)
VALUES ('/mnt/data/rhd/05/Application/0001.pdf', 05, 511, 20000, 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/05/transcript/0001.pdf', 05, 20005,
'bachelor certificate and transcript', 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/05/transcript/0002.pdf', 05, 20005,
'master certificate and transcript', 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/05/financial/0001.pdf',
05, 20010, 30000);

INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
Description, DocStatusID)
VALUES ('/mnt/data/rhd/05/reference/0001.pdf',
05, 511, 20008,
'recommendation letters', 30000);

INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
Description, DocStatusID)
VALUES ('/mnt/data/rhd/05/general/0001.pdf',
05, 511, 20011, 'training certificate', 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/05/faculty_assessment_memo/0001.pdf', 05,
20003, 30000);

-- ---------------------------------------------------------------------------
-- Fwd PhD inquiry.txt
-- 6
SELECT "Populating Applicant 6 info" ;

INSERT INTO Applicant (ApplicantID, FName, LName, Sex, DateAdded,
LastModifiedByStaffID)
VALUES (06, 'Fakhri', 'Bazzaz', 1, '2014-05-05', 1002) ;

INSERT INTO Application (ApplicationID, ApplicantID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUES (611, 06, 40000, '2014-05-06', '2014-05-06', '2014-05-06', 1002, 10000);

-- 100503 Computer Communications Networks
INSERT INTO `Application_Research Area` (ApplicationID, FORCode)
VALUES (611, 100503) ;


INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/06/general/0001.pdf', 06, 20011, 'Award of Degree',
30000);

INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
DocStatusID)
VALUES ('/mnt/data/rhd/06/certificate/0001.pdf', 06, 611, 20004, 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/06/cv/0001.pdf', 06, 20001, 30000) ;

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
```

```
VALUES ('/mnt/data/rhd/06/certificate/0002.pdf', 06, 20004, 'English cert',
30000);

-- add the thesis to documents
INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/06/publication/0001.pdf', 06, 20006, 'masters thesis',
30000);

-- add the transcript to documents
INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/06/transcript/0001.pdf', 06, 20005, 'masters transcript',
30000);

-- add a reference statement as a document
INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
DocStatusID)
VALUES ('/mnt/data/rhd/06/reference/0001.pdf', 06, 611, 20008, 30000) ;

-- add the other reference statement as a document
INSERT INTO Document (UploadLink, ApplicantID, ApplicationID, DocTypeID,
DocStatusID)
VALUES ('/mnt/data/rhd/06/reference/0002.pdf', 06, 611, 20008, 30000) ;



-- ----------------------------------------------------------------------------
-- Fwd Requesting for PhD supervision.txt
-- 7
SELECT "Populating Applicant 7 info" ;

INSERT INTO Applicant (ApplicantID, FName, DateAdded, Email,
LastModifiedByStaffID)
VALUES (07, 'Nemo', '2014-05-07', 'nemo@gmail.com', 1000) ;

INSERT INTO Degree (DegID, ApplicantID, Name, Type, InstitutionName,
InstitutionCountryISOCode)
VALUES (721, 07, 'IT', 'bachelors', 'Manipal University', 'IN') ;

INSERT INTO Degree (DegID, ApplicantID, Name, Type, InstitutionName,
InstitutionCountryISOCode)
VALUES (722, 07, 'Software development and engineering',
'masters', 'West Bengal University', 'IN') ;

INSERT INTO Application (ApplicationID, ApplicantID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUE (711, 07, 40000, '2014-05-07', '2014-05-07', '2014-05-07', 1000, 10000) ;

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, DocStatusID)
VALUES ('/mnt/data/rhd/07/resume/0001.pdf', 07, 20002, 30000);

-- add the publication to documents
INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/07/publication/0001.pptx', 07, 20009, 'Six-plus
presenting a six-element framework for sentiment analysis.pptx', 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/07/publication/0002.docx', 07, 20009, 'The price
prediction framework based upon representation of the opinions', 30000);

INSERT INTO Document (UploadLink, ApplicantID, DocTypeID, Description,
DocStatusID)
VALUES ('/mnt/data/rhd/07/publication/0003.docx', 07, 20009,
'The classification framework for the representation of opinions', 30000);
```

```
-- Denise de Vries flags interest in this application
INSERT INTO `University Staff Member_Application` (StaffID, ApplicationID)
VALUES (1000, 711);

-- John Roddick flags interest in this application
INSERT INTO `University Staff Member_Application` (StaffID, ApplicationID)
VALUES (1002, 711);

-- -----------------------------------------------------------------------------
-- PhD Student.txt
-- 8
SELECT "Populating Applicant 8 info" ;

INSERT INTO Applicant (ApplicantID, FName, Email, DateAdded,
LastModifiedByStaffID)
VALUES (08, 'Sara', 'sara@gmail.com', '2014-05-08', 1003) ;

INSERT INTO Degree (DegID, ApplicantID, Name, Type, YearCompleted, GPA,
InstitutionCountryISOCode)

VALUES (821, 08, 'Masters Comp Sc (Information Security)', 'masters',
'2013-12-31', 6.79, 'MY');

INSERT INTO Application (ApplicationID, ApplicantID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUES (811, 08, 40000, '2014-05-08', '2014-05-08', '2014-05-08', 1003, 10000);

-- some research areas
INSERT INTO `Application_Research Area` (ApplicationID, FORCode)
VALUES (811, 100503) ;

INSERT INTO `Application_Research Area` (ApplicationID, FORCode)
VALUES (811, 89999) ;


-- -----------------------------------------------------------------------------
-- PhD Student1.txt
-- 9
SELECT "Populating Applicant 9 info" ;

INSERT INTO Applicant (ApplicantID, FName, LName, Email, DateAdded,
LastModifiedByStaffID)
VALUES (09, 'Abdul-Allah', 'Al-Sadhan', 'abdul-allah.al-sadhan@gmail.com',
'2014-05-09', 1000) ;

INSERT INTO Application (ApplicationID, ApplicantID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUES (911, 09, 40000, '2014-05-09', '2014-05-09', '2014-05-09', 1000, 10000);

-- -----------------------------------------------------------------------------
-- PhD Student2.txt
-- 10
SELECT "Populating Applicant 10 info" ;

INSERT INTO Applicant (ApplicantID, FName, LName, Email, DateAdded,
LastModifiedByStaffID)
VALUES (10, 'Fahd', 'Al-Hayyan', 'fahd.al-hayyan@ut.edu.sa', '2014-05-10', 1002)
;

INSERT INTO Application (ApplicationID, ApplicantID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUES (1011, 10, 40000, '2014-05-10', '2014-05-10', '2014-05-10', 1002, 10000);

-- -----------------------------------------------------------------------------
-- Request for PhD Supervision.txt
-- 11
SELECT "Populating Applicant 11 info" ;
```

```
INSERT INTO Applicant (ApplicantID, FName, LName, Email, DateAdded,
LastModifiedByStaffID)
VALUES (11, 'Venkatraman', 'Ramakrishnan',
'venkatraman.ramakrishnan@gmail.com', '2014-05-11', 1003);

INSERT INTO Application (ApplicationID, ApplicantID, awardID, DateAdded,
DateLastChecked, DateLastModified, LastModifiedByStaffID, applicationStatusID)
VALUES (1111, 11, 40000, '2014-05-11', '2014-05-11', '2014-05-11', 1003, 10000);

-- Publication (apps)
INSERT INTO `Publication` (PubID, ApplicantID, Title, Abstract, Publication,
IssueNo, IssueDate, OnlineLink, OtherAuthorsNames, Language)
VALUES (2000, 09, "AI in the movies", "This paper details AI portral in the media
and the questions it raise and how they are answered in the litrature
...","Science",7,'5/5/2012',"www.science.com","ronald hume","english" );

-- -------------------------------------------------------------------------------
-- referee (apps)
INSERT INTO `Referee`
(RefID,ApplicationID,Name,Relation,Phone,Email,Profession,AcademicLink,EnglishSpeak
er,EnglishLiterate)
VALUES (1111,111,"james
brown","teacher","012456546","James@brown.com","singer","http://www.linkedin.com/pu
b/James-Brown/11/2d3/a40",1,0);

-- -------------------------------------------------------------------------------
-- Visa (apps)
INSERT INTO `Visa` (VisaID, CountryISOCode, ValidFrom, ValidTo, ApplicantID,
VisaStatusID)
VALUES (3001,'DE','01/01/2013', '30/12/2015', 01, 80002);
```

4.8 Create SQL scripts for required queries

4.9 Monitor and tune the operational system

In the experience of system developers, the populate scripts and queries are highly responsive.

The system will need to be continually monitored in future to ensure additional data does not perversely degrade performance.

As stated above, the indexes chosen and the controlled redundancy should mean that common uses of the database remain responsive.

As the system is only expectd to grow by 6MB a year, we do not anticipate any sudden degradation in performance.

It is important to note that the system has not yet been tested with concurrent user transactions. Apache JMeter can be used to measure performance under simulated concurrent load.

4.10 Update test plan

The test plan has been implemented using the STK Unit testing framework. All user tranactions documented above have been tested using the test data set populated by scripts above. All tests pass.

4.11 Create SQL scripts to test system

```
-- ----------------------------------------------------------------------------
-- The unit tests for RHD DB.
--
-- First, load up the stk_unit utility. This must be done as DB admin. E.g.:
--
--     sudo bash -c \
--       'mysql -h localhost -D rhd  < ../stk_unit1.0-rc6/sql/stk_unit.sql'
--
-- Then, read this file in as root.
-- had to run as system root => we won't be able to run this on the uni system
--
-- Run the test by issuing, as root:
--
--     CALL stk_unit.tc('test_rhd');
--
-- ----------------------------------------------------------------------------

CONNECT mysql ;

DROP DATABASE IF EXISTS test_rhd ;

CREATE DATABASE test_rhd CHARACTER SET = utf8;

CONNECT test_rhd;

DELIMITER $$

-- ensure that our populate scripts put at least some data in every table
CREATE PROCEDURE test_populate_contents()
BEGIN
  DECLARE rowCount INT;
  DECLARE tableName varchar(64) DEFAULT "";
  DECLARE tablesDone INTEGER DEFAULT 0;

  -- iterate over all the tables in RHD DB
  DECLARE curs CURSOR FOR SELECT t.TABLE_NAME from information_schema.tables t
                        WHERE t.TABLE_SCHEMA = 'rhd'
                          AND t.table_type = 'BASE TABLE';

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET tablesDone = 1 ;

  OPEN curs ;

  test_count: LOOP

      FETCH curs INTO tableName ;

      IF tablesDone = 1 THEN
        LEAVE test_count ;
      END IF ;

      SET @s = CONCAT('SELECT COUNT(*) FROM rhd.`', tableName, '` into @OUTVAR') ;
      PREPARE rowCountStmt FROM @s ;
      EXECUTE rowCountStmt;
      SELECT @OUTVAR INTO rowCount;

      CALL stk_unit.assert_false(rowCount = 0, CONCAT('Table has zero rows: ',
                                               tableName)) ;

    END LOOP test_count ;

  CLOSE curs ;

END $$
```

```
CREATE PROCEDURE test_utrans_a1()
BEGIN
  DECLARE appID INT(10);
  DECLARE v_count INT;

  SELECT ApplicantID, COUNT(*)
  FROM rhd.Applicant
  WHERE Applicant.FName = 'Shirin'
    AND Applicant.LName = 'Ebadi'
  INTO appID, v_count ;

  CALL stk_unit.assert_true(
    v_count = 1,
    'Expected exactly one applicant of given name') ;

  CALL stk_unit.assert_true(
    appID = 1,
    'Expected exactly one applicant of given name') ;
END $$

-- CREATE PROCEDURE test_utrans_a2()
-- BEGIN
--   DECLARE pubID INT(10);
--   DECLARE v_count INT;

--   SELECT p.PubID, COUNT(*) FROM rhd.Publication p, rhd.Applicant a
--   WHERE p.ApplicantID = a.ApplicantID
--     AND a.FName = 'Shirin'
--     AND a.LName = 'Ebadi'
--   INTO pubID,

--   SELECT ApplicantID, COUNT(*)
--   FROM rhd.Applicant
--   WHERE Applicant.FName = 'Shirin'
--     AND Applicant.LName = 'Ebadi'
--   INTO appID, v_count ;

--   CALL stk_unit.assert_true(
--     v_count = 1,
--     'Expected exactly one applicant of given name') ;

--   CALL stk_unit.assert_true(
--     appID = 1,
--     'Expected exactly one applicant of given name') ;
-- END $$

CREATE PROCEDURE test_utrans_a3()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_a3_actuals
  SELECT d.DegID FROM rhd.Degree d, rhd.Applicant a
  WHERE d.ApplicantID = a.ApplicantID
    AND a.FName = 'Shirin'
    AND a.LName = 'Ebadi' ;

  -- create a temporary table to store the expected results
  CREATE TEMPORARY TABLE utrans_a3_expecteds (
    DegID int(10) NOT NULL,
    PRIMARY KEY (DegID)
  ) ;
  INSERT INTO utrans_a3_expecteds (DegID)
  VALUES (121), (122);

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_a3_problems
  SELECT problems.DegID
  FROM
  (
      SELECT actuals.DegID
      FROM utrans_a3_actuals actuals
      UNION ALL
      SELECT expecteds.DegID
      FROM utrans_a3_expecteds expecteds
  )  problems
  GROUP BY problems.DegID
  HAVING COUNT(*) = 1
  ORDER BY problems.DegID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_a3_problems',
    'Incorrect results') ;

  DROP TABLE utrans_a3_expecteds ;
  DROP TABLE utrans_a3_actuals ;
  DROP TABLE utrans_a3_problems ;
END $$


-- a4 TODO: need to add visa info

-- ----------------------------------------------------------------------------
-- a5
CREATE PROCEDURE test_utrans_a5()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_a5_actuals
  SELECT d.DocID, dt.Type AS DocType
  FROM rhd.Applicant App
  INNER JOIN rhd.Document d ON d.ApplicantID = App.ApplicantID
  INNER JOIN rhd.`Document Type` dt ON d.DocTypeID = dt.DocTypeID
  WHERE App.FName = 'Shirin'
    AND App.LName = 'Ebadi' ;

  -- create a temporary table to store the expected results
  -- one row, id 131, cv
  CREATE TEMPORARY TABLE utrans_a5_expecteds (
    DocID int(10) NOT NULL,
    DocType varchar(50)
  ) ;
  INSERT INTO utrans_a5_expecteds (DocID, DocType)
  VALUES (131, 'cv');

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_a5_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_a5_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_a5_expecteds expecteds
  )  problems
  GROUP BY problems.DocID
  HAVING COUNT(*) = 1
  ORDER BY problems.DocID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_a5_problems',
    'Incorrect results') ;

  DROP TABLE utrans_a5_expecteds ;
  DROP TABLE utrans_a5_actuals ;
  DROP TABLE utrans_a5_problems ;
END $$


-- --------------------------------------------------------------------------------
-- b) Look up applicant’s applications by applicant name

CREATE PROCEDURE test_utrans_b()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_b_actuals
  SELECT App.ApplicationID, AwType.Type AS awardType
  FROM rhd.Applicant Appt
  INNER JOIN rhd.Application App ON Appt.ApplicantID = App.ApplicantID
  INNER JOIN rhd.`Award Type` AwType ON App.AwardID = AwType.AwardID
  WHERE Appt.FName = 'Shirin'
    AND Appt.LName = 'Ebadi' ;

  -- create a temporary table to store the expected results
  -- expect ApplicationID = 111, Type = 'PhD'
  CREATE TEMPORARY TABLE utrans_b_expecteds (
    ApplicationID int(10) NOT NULL,
    awardType varchar(50)
  ) ;
  INSERT INTO utrans_b_expecteds (ApplicationID, awardType)
  VALUES (111, 'PhD');

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_b_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_b_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_b_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_b_problems',
    'Incorrect results') ;

  DROP TABLE utrans_b_expecteds ;
  DROP TABLE utrans_b_actuals ;
  DROP TABLE utrans_b_problems ;
END $$

-- --------------------------------------------------------------------------------
-- c) Look up applicant’s applications by applicant email
CREATE PROCEDURE test_utrans_c()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_c_actuals
  SELECT App.ApplicationID, AwType.Type AS awardType
  FROM rhd.Applicant Appt
  INNER JOIN rhd.Application App ON Appt.ApplicantID = App.ApplicantID
  INNER JOIN rhd.`Award Type` AwType ON App.AwardID = AwType.AwardID
  WHERE Appt.Email = 'don.memo@hotmail.com' ;

  -- create a temporary table to store the expected results
  -- expect ApplicationID = 211, Type = 'PhD'
  CREATE TEMPORARY TABLE utrans_c_expecteds (
    ApplicationID int(10) NOT NULL,
    awardType varchar(50)
  ) ;
  INSERT INTO utrans_c_expecteds (ApplicationID, awardType)
  VALUES (211, 'PhD');

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_c_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_c_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_c_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_c_problems',
    'Incorrect results') ;

  DROP TABLE utrans_c_expecteds ;
  DROP TABLE utrans_c_actuals ;
  DROP TABLE utrans_c_problems ;
END $$

-- --------------------------------------------------------------------------------
-- d) Look up incomplete applications
CREATE PROCEDURE test_utrans_d()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_d_actuals
  SELECT App.ApplicationID
  FROM rhd.Application App
  WHERE App.applicationStatusID < 10500 ;

  -- create a temporary table to store the expected results
  -- expect all 11 applications
  CREATE TEMPORARY TABLE utrans_d_expecteds (
    ApplicationID int(10) NOT NULL
  ) ;
  INSERT INTO utrans_d_expecteds (ApplicationID)
  VALUES (111), (211), (311), (411), (511), (611), (711), (811), (911), (1011),
  (1111) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_d_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_d_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_d_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_d_problems',
    'Incorrect results') ;

  DROP TABLE utrans_d_expecteds ;
  DROP TABLE utrans_d_actuals ;
  DROP TABLE utrans_d_problems ;
END $$

-- ---------------------------------------------------------------------------
-- e) Look up all correspondences relevant to an application
CREATE PROCEDURE test_utrans_e()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_e_actuals
  SELECT Corr.CorrID
  FROM rhd.Application App
  INNER JOIN rhd.Correspondence Corr ON Corr.ApplicationID = App.ApplicationID
  WHERE App.ApplicationID = 411 ;

  -- create a temporary table to store the expected results
  -- two rows, id 441 and 442,
  CREATE TEMPORARY TABLE utrans_e_expecteds (
    CorrID int(10) NOT NULL
  ) ;
  INSERT INTO utrans_e_expecteds (CorrID)
  VALUES (441), (442) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_e_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_e_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_e_expecteds expecteds
  )  problems
  GROUP BY problems.CorrID
  HAVING COUNT(*) = 1
  ORDER BY problems.CorrID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_e_problems',
    'Incorrect results') ;

  DROP TABLE utrans_e_expecteds ;
  DROP TABLE utrans_e_actuals ;
  DROP TABLE utrans_e_problems ;
END $$
```

```
-- ---------------------------------------------------------------------------
-- g) Look up which staff member updated an Application most recently
CREATE PROCEDURE test_utrans_g()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_g_actuals
  SELECT USM.StaffID, USM.FName, USM.LName
  FROM rhd.Application App
  INNER JOIN rhd.`University Staff Member` USM
    ON USM.StaffID = App.LastModifiedByStaffID
  WHERE App.ApplicationID = 311 ;

  -- create a temporary table to store the expected results
  -- expect 'Paul' 'Calder'
  CREATE TEMPORARY TABLE utrans_g_expecteds (
    StaffID int(10),
    FName varchar(50),
    LName varchar(50)
  ) ;
  INSERT INTO utrans_g_expecteds (StaffID, FName, LName)
  VALUES (1001, 'Paul', 'Calder') ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_g_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_g_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_g_expecteds expecteds
  )  problems
  GROUP BY problems.StaffID
  HAVING COUNT(*) = 1
  ORDER BY problems.StaffID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_g_problems',
    'Incorrect results') ;

  DROP TABLE utrans_g_expecteds ;
  DROP TABLE utrans_g_actuals ;
  DROP TABLE utrans_g_problems ;
END $$

-- ---------------------------------------------------------------------------
-- h) Check for any decision recorded about an application
CREATE PROCEDURE test_utrans_h()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_h_actuals
  SELECT Decision.StaffID, DT.type
  FROM rhd.Application App
  INNER JOIN rhd.Decision ON Decision.ApplicationID = App.ApplicationID
  INNER JOIN rhd.`Decision Type` DT
    ON DT.DecisionTypeID = Decision.DecisionTypeID
  WHERE App.ApplicationID = 411 ;

  -- create a temporary table to store the expected results
  -- expect COUNT = 1
  -- expect StaffID = 1001
  -- expect dectype = 'RFI'
  CREATE TEMPORARY TABLE utrans_h_expecteds (
    StaffID int(10),
    type varchar(50)
  ) ;
  INSERT INTO utrans_h_expecteds (StaffID, type)
  VALUES (1001, 'RFI' ) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_h_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_h_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_h_expecteds expecteds
  )  problems
  GROUP BY problems.StaffID
  HAVING COUNT(*) = 1
  ORDER BY problems.StaffID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_h_problems',
    'Incorrect results') ;

  DROP TABLE utrans_h_expecteds ;
  DROP TABLE utrans_h_actuals ;
  DROP TABLE utrans_h_problems ;
END $$

-- ------------------------------------------------------------------------------
-- k) Look up an existing application and list outstanding information
-- (checklist).
CREATE PROCEDURE test_utrans_k()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_k_actuals
  SELECT App.ApplicationID, AppStat.Status,
    App.AddressConfirmed, App.DegreeConfirmed,
    App.VisaStatusConfirmed, App.ProposalConfirmed, App.HasResearchAreas,
    App.HasPrimarySuper, App.PayMethConfirmed, App.EngProfConfirmed
  FROM rhd.Application App
  INNER JOIN rhd.`Application Status` AppStat
    ON AppStat.ApplicationStatusID = App.ApplicationStatusID
  WHERE App.ApplicationID = 611 ;

  -- create a temporary table to store the expected results
  -- expect count == 1; status=ongoing, others 0/false
  CREATE TEMPORARY TABLE utrans_k_expecteds (
    ApplicationID int(10),
    Status varchar(50),
    AddressConfirmed int(1),
    DegreeConfirmed int(1),
    VisaStatusConfirmed int(1),
    ProposalConfirmed int(1),
    HasResearchAreas int(1),
    HasPrimarySuper int(1),
    PayMethConfirmed int(1),
    EngProfConfirmed int(1)
  ) ;
  INSERT INTO utrans_k_expecteds
  VALUES (611, 'ongoing', 0, 0, 0, 0, 0, 0, 0, 0 ) ;
  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_k_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_k_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_k_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_k_problems',
    'Incorrect results') ;

  DROP TABLE utrans_k_expecteds ;
  DROP TABLE utrans_k_actuals ;
  DROP TABLE utrans_k_problems ;
END $$

-- ---------------------------------------------------------------------------
-- l) Update the checklist to confirm that a mandatory information requirement
-- has been met
CREATE PROCEDURE test_utrans_l()
BEGIN

  UPDATE rhd.Application
  SET AddressConfirmed = 1
  WHERE Application.ApplicationID = 611 ;
  -- expect success
  -- rerun k), expect AddressConfirmed == 1/true

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_l_actuals
  SELECT App.ApplicationID, AppStat.Status,
    App.AddressConfirmed, App.DegreeConfirmed,
    App.VisaStatusConfirmed, App.ProposalConfirmed, App.HasResearchAreas,
    App.HasPrimarySuper, App.PayMethConfirmed, App.EngProfConfirmed
  FROM rhd.Application App
  INNER JOIN rhd.`Application Status` AppStat
    ON AppStat.ApplicationStatusID = App.ApplicationStatusID
  WHERE App.ApplicationID = 611 ;

  -- create a temporary table to store the expected results
  -- expect count == 1; status=ongoing, others 0/false
  CREATE TEMPORARY TABLE utrans_l_expecteds (
    ApplicationID int(10),
    Status varchar(50),
    AddressConfirmed int(1),
    DegreeConfirmed int(1),
    VisaStatusConfirmed int(1),
    ProposalConfirmed int(1),
    HasResearchAreas int(1),
    HasPrimarySuper int(1),
    PayMethConfirmed int(1),
    EngProfConfirmed int(1)
  ) ;
  INSERT INTO utrans_l_expecteds
  VALUES (611, 'ongoing', 1, 0, 0, 0, 0, 0, 0, 0 ) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_l_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_l_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_l_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_l_problems',
    'Incorrect results') ;

  DROP TABLE utrans_l_expecteds ;
  DROP TABLE utrans_l_actuals ;
  DROP TABLE utrans_l_problems ;

  -- reset to initial state
  UPDATE rhd.Application
  SET AddressConfirmed = 1
  WHERE Application.ApplicationID = 611 ;

END $$

-- ---------------------------------------------------------------------------
-- m) Retrieve all on-going applications for which the user has made the most
-- recent correspondence
CREATE PROCEDURE test_utrans_m()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_m_actuals
  SELECT App.ApplicationID
  FROM rhd.Application App
  INNER JOIN rhd.`Application Status` AppStat
    ON AppStat.ApplicationStatusID = App.ApplicationStatusID
  WHERE AppStat.Status LIKE 'ongoing%'
    AND App.LastModifiedByStaffID = 1002 ;

  -- create a temporary table to store the expected results
  -- three rows, ids 111, 611 and 1011
  CREATE TEMPORARY TABLE utrans_m_expecteds (
    ApplicationID int(10)
  ) ;
  INSERT INTO utrans_m_expecteds
  VALUES (111), (611), (1011) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_m_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_m_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_m_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_m_problems',
    'Incorrect results') ;

  DROP TABLE utrans_m_expecteds ;
  DROP TABLE utrans_m_actuals ;
  DROP TABLE utrans_m_problems ;

END $$

-- -----------------------------------------------------------------------------
-- o) Update the status of an application
CREATE PROCEDURE test_utrans_o()
BEGIN

  -- change the status of an application
  UPDATE rhd.Application App
  JOIN
    (SELECT * FROM rhd.`Application Status` AppStat
     WHERE AppStat.Status = 'complete.declined') AS Lookup
  SET App.applicationStatusID = Lookup.ApplicationStatusID
  WHERE App.ApplicationID = 611 ;

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_o_actuals
  SELECT AppStat.Status
  FROM rhd.`Application Status` AppStat
  INNER JOIN rhd.Application App
    ON AppStat.ApplicationStatusID = App.ApplicationStatusID
  WHERE App.ApplicationID = 611;

  -- create a temporary table to store the expected results
  -- one row, 'complete.declined'
  CREATE TEMPORARY TABLE utrans_o_expecteds (
    Status varchar(50)
  ) ;
  INSERT INTO utrans_o_expecteds
  VALUES ('complete.declined');

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_o_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_o_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_o_expecteds expecteds
  )  problems
  GROUP BY problems.Status
  HAVING COUNT(*) = 1
  ORDER BY problems.Status ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_o_problems',
    'Incorrect results') ;

  DROP TABLE utrans_o_expecteds ;
  DROP TABLE utrans_o_actuals ;
  DROP TABLE utrans_o_problems ;

  -- reset the status of an application
  UPDATE rhd.Application App
  JOIN
    (SELECT * FROM rhd.`Application Status` AppStat
     WHERE AppStat.Status = 'ongoing') AS Lookup
  SET App.applicationStatusID = Lookup.ApplicationStatusID
  WHERE App.ApplicationID = 611 ;

END $$

-- ------------------------------------------------------------------------------
-- p) Look up, add to, and delete from own current research areas
-- p1) look up current research areas
CREATE PROCEDURE test_utrans_p1()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_p1_actuals
  SELECT USM.FORCode
  FROM rhd.`University Staff Member_Research Area` USM
  WHERE USM.StaffID = 1000 ;

  -- create a temporary table to store the expected results
  -- expect count == 4, 80399, 80499, 80699, 89999
  CREATE TEMPORARY TABLE utrans_p1_expecteds (
    FORCode int(10)
  ) ;
  INSERT INTO utrans_p1_expecteds
  VALUES  (80399), (80499), (80699), (89999) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_p1_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_p1_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_p1_expecteds expecteds
  )  problems
  GROUP BY problems.FORCode
  HAVING COUNT(*) = 1
  ORDER BY problems.FORCode ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_p1_problems',
    'Incorrect results') ;

  DROP TABLE utrans_p1_expecteds ;
  DROP TABLE utrans_p1_actuals ;
  DROP TABLE utrans_p1_problems ;

END $$

-- p2) Insert new research areas tested in populate script

-- p3) Delete research areas
CREATE PROCEDURE test_utrans_p3()
BEGIN

  -- remove a research area
  DELETE rhd.`University Staff Member_Research Area` USMRA
  FROM rhd.`University Staff Member_Research Area` USMRA
  JOIN
    (SELECT RA.FORCode FROM rhd.`Research Area` RA
     WHERE RA.Description LIKE 'information systems not elsewhere classified')
    AS Lookup
  WHERE StaffID = 1000
    AND USMRA.FORCode = Lookup.FORCode;

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_p3_actuals
  SELECT USM.FORCode
  FROM rhd.`University Staff Member_Research Area` USM
  WHERE USM.StaffID = 1000 ;

  -- create a temporary table to store the expected results
  -- expect count == 4, 80399, 80499, 89999
  CREATE TEMPORARY TABLE utrans_p3_expecteds (
    FORCode int(10)
  ) ;
  INSERT INTO utrans_p3_expecteds
  VALUES  (80399), (80499), (89999) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_p3_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_p3_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_p3_expecteds expecteds
  )  problems
  GROUP BY problems.FORCode
  HAVING COUNT(*) = 1
  ORDER BY problems.FORCode ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_p3_problems',
    'Incorrect results') ;

  DROP TABLE utrans_p3_expecteds ;
  DROP TABLE utrans_p3_actuals ;
  DROP TABLE utrans_p3_problems ;

  -- Re-instate the temporarily removed research area
  INSERT INTO rhd.`University Staff Member_Research Area`
   (StaffID, FORCode)
  (SELECT USM.StaffID, Lookup.FORCode FROM rhd.`University Staff Member` USM
   JOIN
     (SELECT RA.FORCode FROM rhd.`Research Area` RA
      WHERE RA.Description LIKE 'information systems not elsewhere classified')
     AS Lookup
   WHERE USM.StaffID = 1000 );

END $$

-- --------------------------------------------------------------------------------
-- q) Search for all applications in certain research areas that have been added
-- since a certain time
CREATE PROCEDURE test_utrans_q()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_q_actuals
  SELECT App.ApplicationID
  FROM rhd.Application App
  INNER JOIN rhd.`Application_Research Area` ARA
    ON App.ApplicationID = ARA.ApplicationID
  WHERE App.DateAdded >= '2014-05-01'
    AND ARA.FORCode = 100503 ;

  -- create a temporary table to store the expected results
  -- expect 1 row, id 611
  CREATE TEMPORARY TABLE utrans_q_expecteds (
    ApplicationID int(10)
  ) ;
  INSERT INTO utrans_q_expecteds
  VALUES  (611) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_q_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_q_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_q_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_q_problems',
    'Incorrect results') ;

  DROP TABLE utrans_q_expecteds ;
  DROP TABLE utrans_q_actuals ;
  DROP TABLE utrans_q_problems ;

END $$

-- ----------------------------------------------------------------------------
-- s) Retrieve all staff who have flagged an application, or have edited an
-- application or applicant record most recently
--
-- s1) Retrieve the flagged applications
CREATE PROCEDURE test_utrans_s1()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_s1_actuals
  SELECT USM.StaffID, USM.FName, USM.LName
  FROM rhd.`University Staff Member` USM
  INNER JOIN rhd.`University Staff Member_Application` Flags
    ON Flags.StaffID = USM.StaffID
  WHERE Flags.ApplicationID = 711 ;

  -- create a temporary table to store the expected results
  -- expect Denise de Vries and John Roddick
  CREATE TEMPORARY TABLE utrans_s1_expecteds (
    StaffID int(10),
    FName varchar(50),
    LName varchar(50)
  ) ;
  INSERT INTO utrans_s1_expecteds
  VALUES  (1002, 'John', 'Roddick'), (1000, 'Denise', 'de Vries') ;
  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_s1_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_s1_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_s1_expecteds expecteds
  )  problems
  GROUP BY problems.StaffID
  HAVING COUNT(*) = 1
  ORDER BY problems.StaffID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_s1_problems',
    'Incorrect results') ;

  DROP TABLE utrans_s1_expecteds ;
  DROP TABLE utrans_s1_actuals ;
  DROP TABLE utrans_s1_problems ;

END $$

-- s2) Last staff member to modify an application
CREATE PROCEDURE test_utrans_s2()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_s2_actuals
  SELECT USM.StaffID, USM.FName, USM.LName
  FROM rhd.`University Staff Member` USM
  INNER JOIN rhd.Application App
    ON App.LastModifiedByStaffID = USM.StaffID
  WHERE App.ApplicationID = 711 ;

  -- create a temporary table to store the expected results
  -- expect Denise de Vries and John Roddick
  CREATE TEMPORARY TABLE utrans_s2_expecteds (
    StaffID int(10),
    FName varchar(50),
    LName varchar(50)
  ) ;
  INSERT INTO utrans_s2_expecteds
  VALUES  (1000, 'Denise', 'de Vries') ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_s2_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_s2_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_s2_expecteds expecteds
  )  problems
  GROUP BY problems.StaffID
  HAVING COUNT(*) = 1
  ORDER BY problems.StaffID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_s2_problems',
    'Incorrect results') ;

  DROP TABLE utrans_s2_expecteds ;
  DROP TABLE utrans_s2_actuals ;
  DROP TABLE utrans_s2_problems ;

END $$

-- ------------------------------------------------------------------------------
-- t) Retrieve all ongoing applications
CREATE PROCEDURE test_utrans_t()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_t_actuals
  SELECT App.ApplicationID
  FROM rhd.Application App
  INNER JOIN rhd.`Application Status` AppStat
    ON App.ApplicationStatusID = AppStat.ApplicationStatusID
  WHERE AppStat.Status LIKE 'ongoing%' ;

  -- create a temporary table to store the expected results
  -- expect all applications at this stage
  CREATE TEMPORARY TABLE utrans_t_expecteds (
    ApplicationID int(10)
  ) ;
  INSERT INTO utrans_t_expecteds
  VALUES (111), (211), (311), (411), (511), (611), (711), (811), (911), (1011),
  (1111) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_t_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_t_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_t_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_t_problems',
    'Incorrect results') ;

  DROP TABLE utrans_t_expecteds ;
  DROP TABLE utrans_t_actuals ;
  DROP TABLE utrans_t_problems ;

END $$

-- ------------------------------------------------------------------------------
-- u) List all applications waiting for supervisor agreement
CREATE PROCEDURE test_utrans_u()
BEGIN

  -- run the query for this test and create a temporary table for results
  CREATE TEMPORARY TABLE utrans_u_actuals
  SELECT Agg.ApplicationID FROM
    (SELECT
       App.ApplicationID,
       Super.StaffID,
       Super.PrimarySupervisor,
       SUM(Super.PrimarySupervisor) AS PriSuperSum
     FROM rhd.Application App
     LEFT OUTER JOIN rhd.`Supervise as` Super
       ON Super.ApplicationID = App.ApplicationID
     GROUP BY App.ApplicationID)
    AS Agg
  WHERE Agg.PriSuperSum <> 1
    OR Agg.PriSuperSum IS NULL ;

  -- create a temporary table to store the expected results
  -- expect all applications except 111 and 311
  CREATE TEMPORARY TABLE utrans_u_expecteds (
    ApplicationID int(10)
  ) ;
  INSERT INTO utrans_u_expecteds
  VALUES (211), (411), (511), (611), (711), (811), (911), (1011), (1111) ;

  -- create another temporary table that lists the differences between expected
  -- results and actuals
  CREATE TEMPORARY TABLE utrans_u_problems
  SELECT problems.*
  FROM
  (
      SELECT actuals.*
      FROM utrans_u_actuals actuals
      UNION ALL
      SELECT expecteds.*
      FROM utrans_u_expecteds expecteds
  )  problems
  GROUP BY problems.ApplicationID
  HAVING COUNT(*) = 1
  ORDER BY problems.ApplicationID ;

  -- report an error if the 'problems' table isn't empty
  CALL stk_unit.assert_table_empty(
    DATABASE(),
    'utrans_u_problems',
    'Incorrect results') ;

  DROP TABLE utrans_u_expecteds ;
  DROP TABLE utrans_u_actuals ;
  DROP TABLE utrans_u_problems ;

END $$


DELIMITER ;
```

4.12 Test operational system

We ran the test code listed above in section 4.11 using the STK (Picchiarelli and Razzoli 2013). Each of these tests defines the query to run and the expected results. It then automatically compares the actual results with the expected results and logs any differences. Below is a report of running these tests. Note that due to the restrictions of the university system, we ran these tests on other MySQL servers with which we had more control.

```
+-------------------------------------------------------------------------------+
| report                                                                        |
+-------------------------------------------------------------------------------+
|
Test Case: test_rhd
Id: 25
Completed: YES
46 passes, 0 fails, 0 exceptions
 |
+-------------------------------------------------------------------------------+
```

4.13 References

Australian Bureau of Statistics (2014). "Field of Research Codes." from http://www.abs.gov.au/Ausstats/abs@.nsf/Latestproducts/6BB427AB9696C225CA2574180004463E?opendocument.

Flinders University Office of Graduate Research (2014a). "Find a supervisor." Retrieved 27 Mar 2014, from http://www.flinders.edu.au/graduate-research/find-a-supervisor.cfm.

Flinders University Office of Graduate Research (2014b). "How to apply." from http://www.flinders.edu.au/graduate-research/future-students/how-to-apply.cfm.

Oracle Inc. (2014a). "MySQL 5.1 Reference Manual - 1.8.2 MySQL Differences from Standard SQL." Retrieved 12/06/2014, from http://dev.mysql.com/doc/refman/5.1/en/differences-from-ansi.html.

Oracle Inc. (2014b). "MySQL 5.1 Reference Manual - 13.1.17 CREATE TABLE Syntax." Retrieved 12/06/2014, from http://dev.mysql.com/doc/refman/5.1/en/create-table.html.

Oracle Inc. (2014c). "MySQL 5.1 Reference Manual - 14 Storage Engines." Retrieved 11/6/2014, from http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html.

Oracle Inc. (2014d). "MySQL 5.1 Reference Manual - 14.6.3.12 InnoDB Table and Index Structures." Retrieved 11/06/2014, from http://dev.mysql.com/doc/refman/5.1/en/innodb-table-and-index.html.

Picchiarelli, G. and F. Razzoli (2013). "STK/Unit." Retrieved 2014.04.09, from http://stk.wikidot.com/stk-unit.

Wikipedia (2014a). "Academic Grading in Australia." Retrieved 27 Mar 2014, from http://en.wikipedia.org/wiki/Academic_grading_in_Australia#Grade_point_average.

Wikipedia (2014b). "Grade Point Average." Retrieved 27 Mar 2014, from http://en.wikipedia.org/wiki/GPA.

5 Data dictionary