Daniel Patrick(CS), Lauren Vaughn(CS)

Assignment 3

25 October 2017

**Logical Database Design**

The purpose is to develop a relational data model based upon the ER Model developed during the Conceptual Data Base Design phase. The ER Diagram is shown in figure 1 as reference.

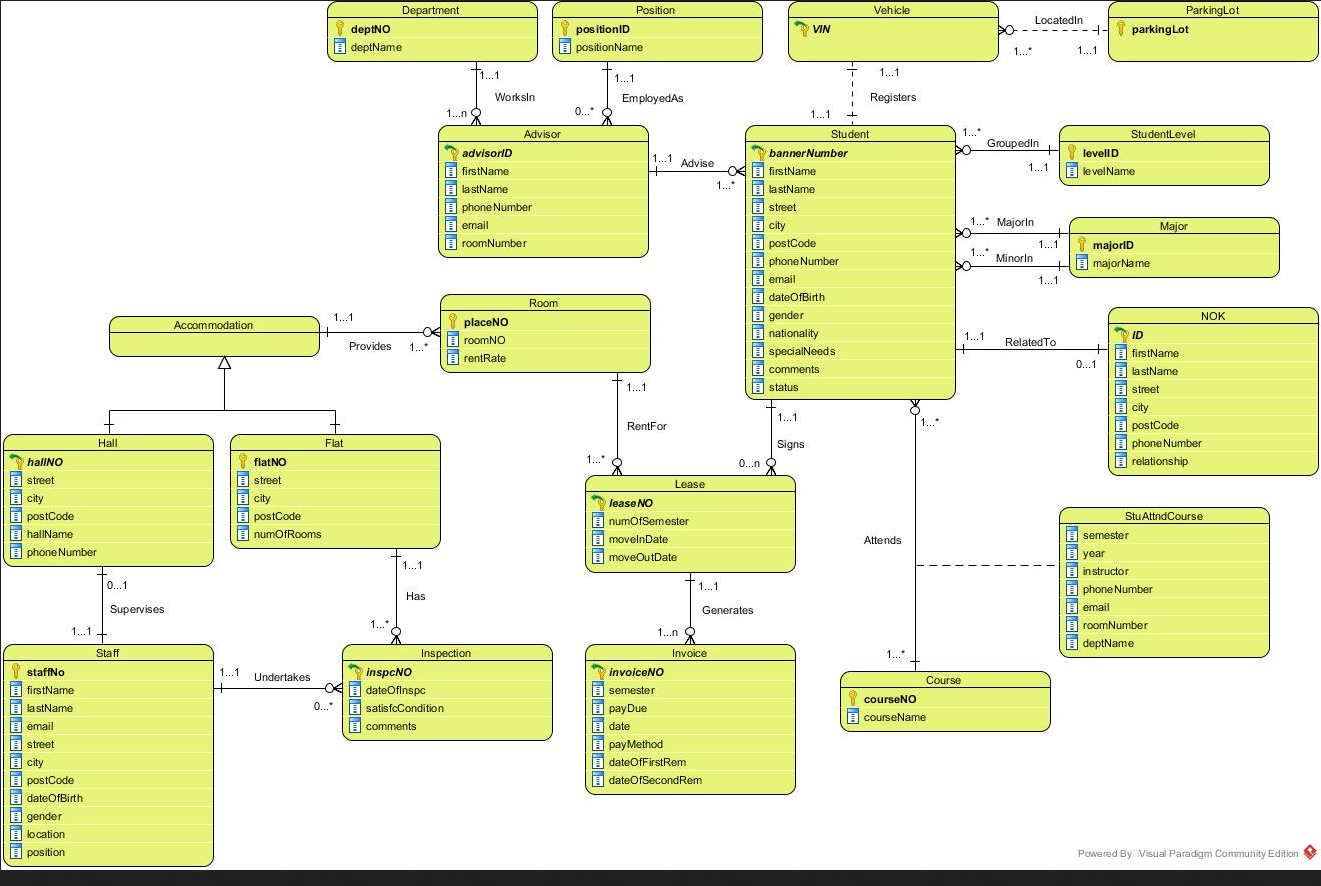


Figure : ER Diagram for MU Housing

Standard conversion rules were used to translate this ER Diagram into an equivalent relational data model.

**Initial Relational Data Model:**

**Please note**: Primary keys and foreign key attributes are identified as follows: Primary key(s) are underlined and foreign keys are *italicized*.

StudentInfo (bannerNumber, *levelID*, *majorID*, *advisorID*, *parkingLot,* firstName, lastName, street, city, postcode, phoneNumber, email, dateOfBirth, gender, nationality, specialNeeds, comments, status, vin, color, manufacturer, brand)

ParkingLot (parkingLot, lotName, address, maxSpace, availability)

StudentLevel (levelID,levelName)

Major (majorID,majorName)

NOK (ID, *bannerNumber*, firstName, lastName, street, city, phoneNumber, relationship)

StudentCourse( bannerNumber,courseNO**,** semester, year, instructor, phoneNumber, email, roomNumber, deptName)

Course (courseNo, courseName)

Lease( leaseNO, *bannerNO*, *placeNO*, numOfSemester, moveInDate, moveOutDate)

Invoice (invoiceNo, *leaseNO*, semester, payDue, date, payMethod, dateOfFirstRem, dateOfSecondRem)

Room( placeNO, *flatNO*, *hallNO*, roomNO, rentRate)

Hall (hallNO, *staffNo*, street, city, postcode, hallName, phoneNumber)

Flat (flatNO, street, city, postcode, numOfRooms)

Staff (staffNo, firstName, lastName, email, street, city, postcode, dateOfBirth, gender, location, position )

Inspection(inspcNo, *staffNO*, *flatNO*, dateOfInspc, satisfcCondition, comments)

Advisor (advisorID, *positionID*, *deptNO*, firstName, lastName, phoneNumber, email, roomNumber)

Position (positionID, postionName)

Department (deptNO, deptName)

**Definition of Attributes**

Here the meaning of each attribute and relation is defined. The MU Housing Model includes 17 relations.

The relation StudentInfo is used to store information about Students. Each row (n-tuple or simply tuple) in the relation corresponds to one Student in the real world. Thousands of students can reside in database.

**bannerNumber**. is the primary key for Student. It contains a unique value for each student. It takes alphanumeric values.

**firstName**. It is an attribute used to store the first name of a student. It contains alphanumeric data and should be large enough to hold 30 characters.

**lastName.** It is an attribute used to store the last name of a student. It contains alphanumeric data. It contains alphanumeric data and should be large enough to hold 30 characters.

**street.** It is an attribute used to store the home street name of a student. It contains alphanumeric data and is large enough to hold up to 50 characters.

**city.** It is an attribute used to store the home city name of a student. It contains alphanumeric data and is large enough to hold up to 30 characters.

**postcode**. It is an attribute used to store the home postal code of a student. It contains numeric data and holds 5 characters.

**phoneNumber.** It is an attribute used to store the home phone number of a student. It contains numeric data.

**email**. It is an attribute used to store the email of a student. It contains alphanumeric data and is large enough to hold 30 characters.

**dateOfBirth**. It is an attribute used to store the date of birth of a student. It contains alpha numeric data and is large enough to hold 10 characters.

**gender**. It is an attribute used to store the gender of a student. It contains alphanumeric data.

**nationality**. It is an attribute used to store the nationality of a student. It contains alphanumeric data.

**specialNeeds**. It is an attribute used to store the special needs of a student. It contains alphanumeric data.

**comments**. It is an attribute used to store the comments of a student. It contains alphanumeric data.

**status**. It is an attribute used to store the status of a student. It contains alphanumeric data.

**vin.** An attribute used to store student vehicle vin number. Contains alphanumeric data and is large enough to hold 17 digits.

**color**. An attribute used to store student vehicle color. Contains alphanumeric data.

**manufacturer**. Attribute used to store student vehicle manufacturer. Contains alphanumeric data.

**brand**. Attribute used to store student vehicle brand. Contains alphanumeric data.

The relation ParkingLot stores basic information about parking lots. Each row (n-tuple or simply tuple) in the relation corresponds to one residence hall parking lot.

**parkingLot.** Acts as a primary key for ParkingLot. Contains a uniquely-identifying number for each parking lot.

**lotName.** Attribute containing the parking lot name. Contains an alphanumeric value, allow for 20 characters in length.

The relation StudentLevel stores information about possible student levels (freshman-5 year senior, graduate). Each row (n-tuple or simply tuple) in the relation corresponds to one student level.

**levelID**. This is the primary key for StudentLevel. It contains a unique value for each student level. It takes numeric values.

**levelName**. It is an attribute used to store the level name of a StudentLevel. It contains alphanumeric data.

The relation Major is used to store information about Student Major. Each row (n-tuple or simply tuple) in the relation corresponds to one Major. There are a maximum of 100 majors.

**majorID**. This attribute is the primary key for Major. It contains a unique value for each major. It takes numeric values.

**majorName**. It is an attribute used to store the major name of a Major. It contains alphanumeric data.

The relation NOK is used to store information about Students’ next of kin . Each row (n-tuple or simply tuple) in the relation corresponds to one next of kin. There are many thousand next of kin.

**ID**. is the primary key for NOK. It contains a unique value for each next-of-kin. It takes numeric values.

**firstName**. It is an attribute used to store the first name of a next of kin. It contains alphanumeric data.

**lastName**. It is an attribute used to store the last name of a next of kin. It contains alphanumeric data.

**street.** It is an attribute used to store the home street name of a next of kin. It contains alphanumeric data and should be large enough to hold up to 50 characters.

**city.** It is an attribute used to store the home city name of a next of kin. It contains alphanumeric data and should be large enough to hold up to 30 characters.

**phoneNumber.** It is an attribute used to store the home phone of a next of kin. It contains alphanumeric data.

**relationship**. It is an attribute used to store the relationship of a next of kin. It contains alphanumeric data.

The StudentCourse relation stores information about which students have taken which courses. It has a row for each combination of a student and a course. The primary key of this relation contains two attributes bannerNumber and courseNO.

**semester**. It is an attribute used to store the semester a courses is offered. It contains alphanumeric data.

**year**. It is an attribute used to store the year a courses is offered. It contains alphanumeric data.

**instructor.** An attribute to hold the name of a course instructor. Contains alphanumeric data and should be large enough to hold 50 characters.

**phoneNumber.** It is an attribute used to store the on-campus phone number of an instructor. It contains alphanumeric data.

**email**. It is an attribute used to store the email of an instructor. It contains alphanumeric data and is large enough to hold 30 characters.

**roomNumber.** It is an attribute used to store the room number of an instructor. It contains alphanumeric data and is large enough to hold 30 characters.

**deptName.** This is an attribute that holds course department names. Can contain alphanumeric values and should be large enough to hold 20 characters.

The relation Course is used to store information about Courses . Each row (n-tuple or simply tuple) in the relation corresponds to one course.

**courseNo**. It is the primary key for Course. It contains a unique value for each course. It takes alphanumeric values.

**courseName**. It is an attribute used to store the course name of a course It contains alphanumeric data.

The relation Lease is used to store information about student leases. Each row (n-tuple or simply tuple) in the relation corresponds to one student lease.

**leaseNO**. It is the primary key for lease. It contains a unique value for each lease. It takes alphanumeric values.

**numOfSemester**. It is an attribute used to store the number of a semester. It contains alphanumeric data.

**moveInDate**. It is an attribute used to store the move in date. It contains alphanumeric data.

**moveOutDate**. It is an attribute used to store the move in date. It contains alphanumeric data.

The relation Invoice is used to store information about student invoices. Each row (n-tuple or simply tuple) in the relation corresponds to one student invoice.

**invoiceNo**. It is the primary key for Invoice. It contains a unique value for each invoice. It takes alphanumeric values.

**semester**. It is an attribute used to store the semester of a lease. It contains alphanumeric data.

**payDue.** It is an attribute used to store the due date of a payment. It contains alphanumeric data.

**date.** It is an attribute used to store the date of a payment. It contains alphanumeric data.

**payMethod**. It is an attribute used to store the method of a payment. The payment method should be protected from unauthorized access. It holds alphanumeric characters. It is stored as an encrypted data.

**dateOfFirstRem**. It is an attribute used to store the first reminder date of a payment. It contains alphanumeric data.

**dateOfSecondRem**. It is an attribute used to store the second reminder date of a payment. It contains alphanumeric data.

The relation Room is used to store information about each room in student housing. Each row (n-tuple or simply tuple) in the relation corresponds to one room.

**placeNO.** This is the primary key for Room. This attribute holds the unique identifier for each room (residence halls and flats) and contains alphanumeric data.

**roomNO.** Attribute contains the room number for each room (whether is res hall or flats). Contains alphanumeric data.

**rentRate.** This attribute holds a dollar value indicating the amount (in dollars) that monthly rent is for a room.

The relation Hall is used to store information about each residence hall. Each row (n-tuple or simply tuple) in the relation corresponds to a particular residence hall.

**hallNO.** This is the primary key for Hall. Holds the unique number for each residence hall, that is, can take alphanumeric values.

**street.** Attribute contains street location of residence hall. Should be large enough to hold up to 50 alphanumeric characters.

**city.** Holds city location of residence hall. May contain only alphanumeric data and should be large enough to hold up to 20 alphanumeric characters.

**postCode.** Attribute contains postal code of residence hall. May take alphanumeric values.

**hallName.** Attribute holds the name of residence Hall. May contain only alphanumeric data and can hold up to 20 characters.

**phoneNumber.** Used to store residence hall phone number. May contain only alphanumeric data.

The relation Flat is used to store information about Student flats. Each row (n-tuple or simply tuple) in the relation corresponds to one flat. There are 300 flats.

**flatNO.** Acts as primary key for Flat. May contain only alphanumeric data.

**street.** It is an attribute used to store the street address of a flat. It contains alphanumeric data and should be large enough to hold up to 50 characters.

**city.** It is an attribute used to store city name of a flat. It contains alphanumeric data and should be large enough to hold up to 15 characters.

**postCode.** This attribute contains postal code of flat. May take alphanumeric values.

**numOfRooms.** An attribute that holds the number of individual rooms in a flat. Should be 1 number that is at least 1 and less than or equal to 5.

The relation Staff is used to store information about residence hall staff. Each row (n-tuple or simply tuple) in the relation corresponds to one employee.

**staffNo.** This is the primary key for Staff. It contains a unique value for each next-of-kin. It takes numeric values and can store up to 10 digits.

**firstName**. It is an attribute used to store the first name of an employee. It contains alphanumeric data and should be large enough to hold 30 characters.

**lastName.** It is an attribute used to store the last name of an employee. It contains alphanumeric data and should be large enough to hold 30 characters.

**email**. It is an attribute used to store the email of an employee. It contains alphanumeric data and is large enough to hold 30 characters.

**street.** It is an attribute used to store the home street name of an employee. It contains alphanumeric data and is large enough to hold up to 50 characters.

**city.** It is an attribute used to store the home city name of an employee. It contains alphanumeric data and is large enough to hold up to 30 characters.

**postcode**. It is an attribute used to store the home postal code of an employee. It contains alphanumeric data.

**phoneNumber.** It is an attribute used to store the home phone number of an employee. It contains alphanumeric data.

**dateOfBirth**. It is an attribute used to store the date of birth of a student. It contains alpha numeric data and is large enough to hold 10 characters.

**gender**. It is an attribute used to store the gender of an employee. It contains alphanumeric data.

**location.** An attribute used to store the work-location information about an employee. It contains alphanumeric data and is large enough to hold 15 characters.

**position.** An attribute used to store the work-position information about an employee. It contains alphanumeric data and is large enough to hold 15 characters.

The relation Inspection is used to store information about flat inspections. Each row (n-tuple or simply tuple) in the relation corresponds to one flat inspection.

**inspcNo.** It is the primary key for Inspection. It contains a unique value for each inspection and contains alphanumeric values.

**dateOfInspc**. It is an attribute used to store the date of an inspection. It contains alpha numeric data and is large enough to hold 10 characters.

**satisfcCondition**. This attribute is used to store whether the flat was in satisfactory condition (the outcome of the inspection). It contains alphanumeric data.

**comments.** This attribute is used to store any additional comments that were a result of the inspection. Should be large enough to hold 100 characters and contains alphanumeric values.

The relation Advisor is used to store information about student advisors. Each row (n-tuple or simply tuple) in the relation corresponds to one flat inspection.

**advisorID.** This attribute holds the unique number for each advisor and is the primary key for Advisor. Contains alphanumeric values.

**firstName**. It is an attribute used to store the first name of an advisor. It contains alphanumeric data and should be large enough to hold 30 characters.

**lastName.** It is an attribute used to store the last name of an advisor. It contains alphanumeric data. It contains alphanumeric data and should be large enough to hold 30 characters.

**phoneNumber.** It is an attribute used to store the on-campus phone number of an advisor. It contains alphanumeric data.

**email**. It is an attribute used to store the email of an advisor. It contains alphanumeric data and is large enough to hold 30 characters.

**roomNumber.** It is an attribute used to store the room number of an advisor. It contains alphanumeric data and is large enough to hold 30 characters.

The relation Position is used to store information about Positions held at the university. Each row (n-tuple or simply tuple) in the relation corresponds to one position.

**positionID**. This attribute holds the unique ID for each position and is the primary key for Position. Contains alphanumeric values.

**positionName**. This is an attribute that holds position names. Can contain alphanumeric values and should be large enough to hold 20 characters.

Department (deptNO, deptName)

The relation Department is used to store information about departments at the university. Each row (n-tuple or simply tuple) in the relation corresponds to one department.

**deptID.** This attribute holds the unique ID for each department and is the primary key for department. Contains alphanumeric values.

**deptName.** This is an attribute that holds department names. Can contain alphanumeric values and should be large enough to hold 20 characters.

Foreign key attribute definitions are the same as their corresponding referencing attributes that are defined above.

Now, must undergo the normalization process for further model verification.

**Validation of Relational Data Model**

In this step, the given user queries are cross-checked against the relational data model to ensure that the appropriate attributes and relations have been included in order to support the queries and any other requirements. This step will check each of the fifteen queries that the client has included.

Query #1. Present a report listing the Manager’s name and telephone number for each hall of residence.

**Given:** Residence Hall(s).

**Relations:** Hall (hallNO, *staffNo*, hallName, phoneNumber)

Staff (staffNo, position)

Query #2. Present a report listing the names and MU numbers of students with the details of their lease agreements.

**Given:** No particular value. Just list student details.

**Relations:** Student (bannerNumber, firstName, lastName)

Lease ( leaseNO, *bannerNO*, numOfSemester, moveInDate, moveOutDate)

Query #3. Display the details of lease agreements that include the summer semester.

**Given**: A lease includes the summer semester.

**Relations**: Lease( leaseNO, *bannerNO*, numOfSemester, moveInDate, moveOutDate)

Query #4. Display the details of the total rent paid by a given student.

**Given**: A particular student.

**Relations**: Student (bannerNumber, firstName, lastName)

Lease( leaseNO, *bannerNO*)

Invoice (invoiceNo, *leaseNO*, payDue, date, payMethod)

Query #5. Present a report on students who have not paid their invoices by a given date.

**Given:** Student invoice is not paid by a certain date.

**Relations:** Student (bannerNumber, firstName, lastName)

Lease( leaseNO, *bannerNO*)

Invoice (invoiceNo, *leaseNO*, payDue)

Query #6. Display the details of apartment inspections where the property was found to be in an unsatisfactory condition.

**Given:** An apartment is found to be unsatisfactory.

**Relations:** Inspection(inspcNo, *staffNO*, *flatNO*, dateOfInspc, satisfcCondition, comments)

Query #7. Present a report of the names and MU numbers of students with their room number and place number in a particular hall of residence.

**Given:** No particular value, just list students.

**Relations:** Student (bannerNumber, firstName, lastName)

Lease( leaseNO, *bannerNumber, placeNO*)

Room( placeNO, *hallNO*, roomNO)

Hall (hallNO, hallName)

Query #8. Present a report listing the details of all students currently on the waiting list for accommodation; that is; who were not places.

**Given:** A student is on the waiting list.

**Relations :** Student (bannerNumber, firstName, lastName, street, city, postcode, phoneNumber, email, dateOfBirth, gender, nationality, specialNeeds, comments, status)

Query #9. Display the total number of students in each student category.

**Given:** No particular value. Just need to list students.

**Relations:** Student (bannerNumber, *levelID*)

StudentLevel (levelID,levelName)

Query #10. Present a report of the names and MU numbers for all students who have not supplied details of their next-of-kin.

**Given:** A student has not given details for next of kin.

**Relations:** Student (bannerNumber, firstName, lastName)

NOK (ID, *bannerNumber*)

Query #11. Display the name and internal telephone number of the Advisor for a particular student.

**Given:** A particular student.

Relations: Student (bannerNumber, *advisorID*)

Advisor (advisorID, firstName, lastName, phoneNumber)

Query #12. Display the minimum, maximum, and average monthly rent for rooms in residence halls.

**Given:** Residence hall(s).

**Relations:** Room( placeNO, *hallNO* , rentRate)

Hall (hallNO, hallName)

Query #13. Display the total number of places in each residence hall.

**Given:** Residence Halls.

**Relations:** Room( placeNO, *hallNO*)

Hall (hallNO, hallName)

Query #14. Display the staff number, name, age, and current location of all members of the residence staff who are over 60 years old today.

**Given:** Residence hall staff member is over 60 years old today.

**Relations:** Staff (staffNo, firstName, lastName, email, street, city, postcode, dateOfBirth)

Query #15. Display the total number of registered vehicles in the particular parking lot.

**Given**: A particular Parking lot.

**Relations**: StudentVehicle (vin, *parkingLot*)

ParkingLot (parkingLot, lotName)

As desired, the given data model has the appropriate attributes and relations to satisfy the given queries.

**Non-Functional Requirements**

**Volume:**

* # of students = 6,500
* # staff = 300
* # Advisors = 200
* # Majors = 100
* # Minors = 100
* # StudentLevels = 6
* # Departments = 25
* # Rooms = 6,000
* # Flats = 300
* # Residence Halls = 5
* # Parking Lots = 5

**Integrity Constraints:**

The number of rooms in a flat must be at least 1 and less than or equal to 5.

Rent rate cannot be less than $350.00 and not greater than $800.00.

A vin number must be seventeen characters in length (or not valid).

All dates are formatted as mm/dd/yyyy.