FACULTY OF INFORMATION TECHNOLOGY

DATABASE SYSTEMS 600

1ST SEMESTER ASSIGNMENT

Name & Surname: Tawana Kombora **ICAS/ITS No:** 402202621

Qualification: BSc I.T **Semester:** 1ST **Module Name:** Database Systems 600

Submission Date: 31st May 2023

ASSESSMENT CRITERIA	MARK ALLOCATION	EXAMINER MARKS	MODERATOR MARKS
MARKS FOR CONTENT			
QUESTION ONE	**40**		
QUESTION TWO	**30**		
QUESTION THREE	**20**		
TOTAL MARKS	**90**		
MARKS FOR TECHNICAL ASPECTS			
1. TABLE OF CONTENTS Accurate numbering according to the numbering in text and page numbers.	**2**		
2. LAYOUT AND SPELLING Font – Calibri 12 Line Spacing – 1.0 Margin should be justified.	**3**		
3. REFERENCE According to the Harvard Method	**5**		
TOTAL	**10**		
TOTAL MARKS FOR ASSIGNMENT	**100**		
Examiner's Comments:			
Moderator's Comments:			
Signature of Examiner:	**Signature of Moderator:**		

Table of Contents

QUESTION ONE

Question 1.1

A functional dependency in a relational database is a connection between two attributes (or columns) where the value of one attribute affects the value of the other.

This idea is crucial for database normalization, which entails arranging a table's attributes to get rid of duplication and other potential problems.

Transitive Dependency: In a transitive dependency, one attribute determines another indirectly through a third attribute.

For example, consider the following table:

Customer Name	Home Address	City	Province

The column "Customer Name" in this table determines the "Home Address," and the "Home Address" determines the "City" and "Province." This is a transitive dependency because "City" & "Province" are indirectly dependent on "Customer Name" through "Home Address."

We would need to separate this data into two tables—one for customer information and one for address information—in order to standardize it.

By doing this, the transitive dependency would be removed and each table's primary key would be unique.

<>

A non-key attribute that depends on the complete main key, rather than only a portion of it, is referred to as having a full functional dependency.

<>

For example, consider the following table: Student ID | Course | Grade

The primary key in this table is "Student ID"; the non-key attributes "Course" and "Grade" are not keys.

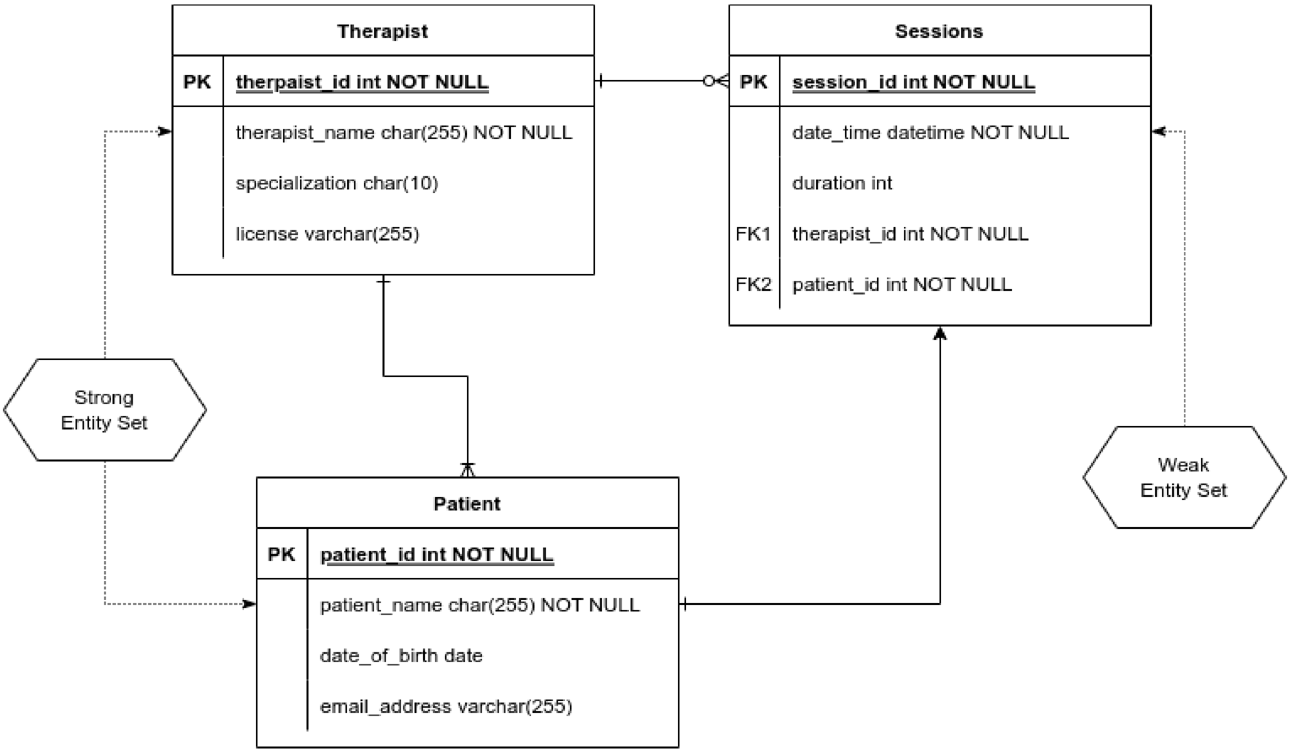
We have a partial dependency if "Grade" depends solely on "Course" and not on the "Student ID." However, if "Grade" is reliant on both "Student ID" and "Course," we have a complete functional dependency.

For instance, imagine that each student is only permitted to enroll in a course once, making each "Student ID" and "Course" combination distinct.

As a result, “Grade” would be entirely reliant on the two attributes. If this data were to be normalized, it would be divided into two separate tables: one for student information and one for course information, with a third table for grades that uses the “Student ID” and “Course” attributes as a composite primary key.

Question 1.2

- Strong-Entity set: a set that doesn’t depend on other entities to exist. It is independent.
- Weak-Entity set: a set that depends on another entity to exist.



QUESTION TWO

Question 2.1

A view will stop working if one of the tables it depends on is dropped. The reason for this is that a view is a virtual table (which is to say a view is calculated) built from underlying tables, any changes to those tables will have an impact on the calculation thus changing the view.

The database system will produce an error indicating that the view cannot be found or that it is invalid if the view is accessed after the underlying table has been dropped. This is due to the dependence on the missing table.

The view will need to be changed or removed in order to fix this problem. The view can simply be dropped if it is no longer required.

Views are stored in the database as *metadata* rather than as actual *physical tables*. Dropping a table being used by a view won't change the content of the other physical tables' actual data, but it will change how those tables may be accessed by the view.

Question 2.2

Hadoop is used to store and handle massive distributed data collections. It is made to handle huge data workloads that conventional relational database management systems are unable to handle efficiently. Hadoop is a great option for businesses working with big volumes of data because of its high scalability and capacity to manage petabytes of data.

Hadoop processes big data sets over a number of devices using a distributed file system (HDFS) and a distributed computation architecture (MapReduce).

Cost-effectiveness: Hadoop is open-source and free to use, which makes it a cost-effective solution for big data processing.

Flexibility: Hadoop supports a wide range of data formats, including structured, semi-structured, and unstructured data. This makes it easier to work with diverse data sets that may not fit neatly into a traditional relational database. Hadoop is very adaptable and can be expanded with new tools and libraries to fulfill specialized business requirements

SPSS

SPSS is built for smaller data sets and could have trouble with large amounts of data, normally it operates on a single machine, this enables quicker processing times and more fault tolerance.

SPSS requires a license fee, which can be a barrier for smaller organizations or those with limited budgets. It is primarily designed for structured data and may struggle with other data formats. It is a closed system that might be difficult to extend.

Hadoop is a superior option over SPSS for large data pre-processing. Hadoop is better suited for big data workloads that demand processing and analysis of large-scale distributed data sets, whereas SPSS may be a preferable option for smaller data sets or specialised analytical tasks.

QUESTION THREE

Question 3.1

Students entity set:

Primary-key: Student ID

Attributes: Student ID, Name, Gender, Date of Birth, Contact Details (Phone Number, Email Address), Address, Nationality, Enrollment Date

Lecturers entity set:

Primary-key: Lecturer ID

Attributes: Lecturer ID, Name, Gender, Date of Birth, Contact Details (Phone Number, Email Address), Address, Nationality, Department ID

Courses entity set:

Primary-key: Course ID

Attributes: Course ID, Course Name, Course Description, Department ID

Modules entity set:

Primary-key: Module ID

Attributes: Module ID, Module Name, Module Description, Course ID, Lecturer ID

Departments entity set:

Primary-key: Department ID

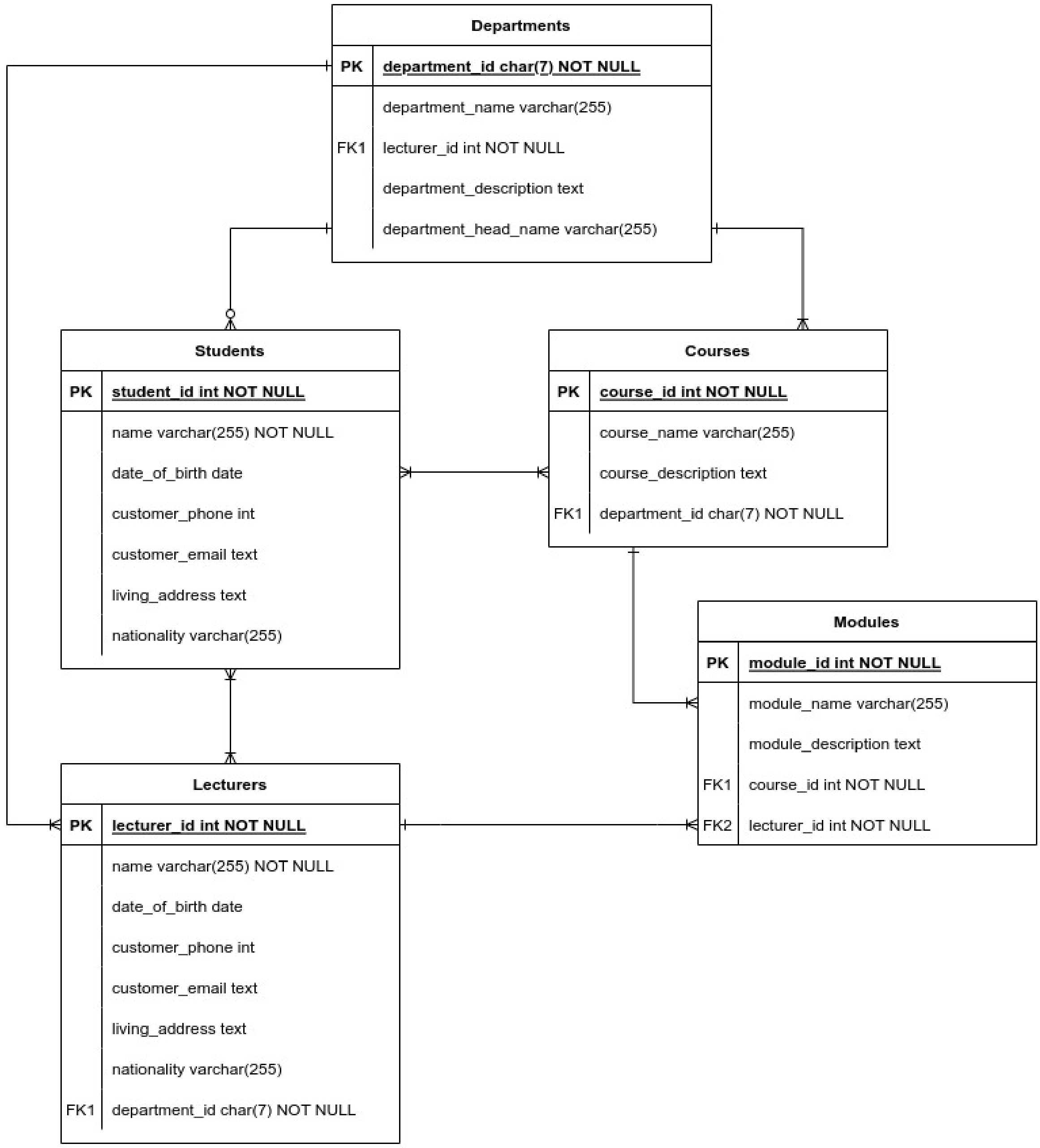
Attributes: Department ID, Department Name, Department Head (Lecturer ID), Department Description

Question 3.2

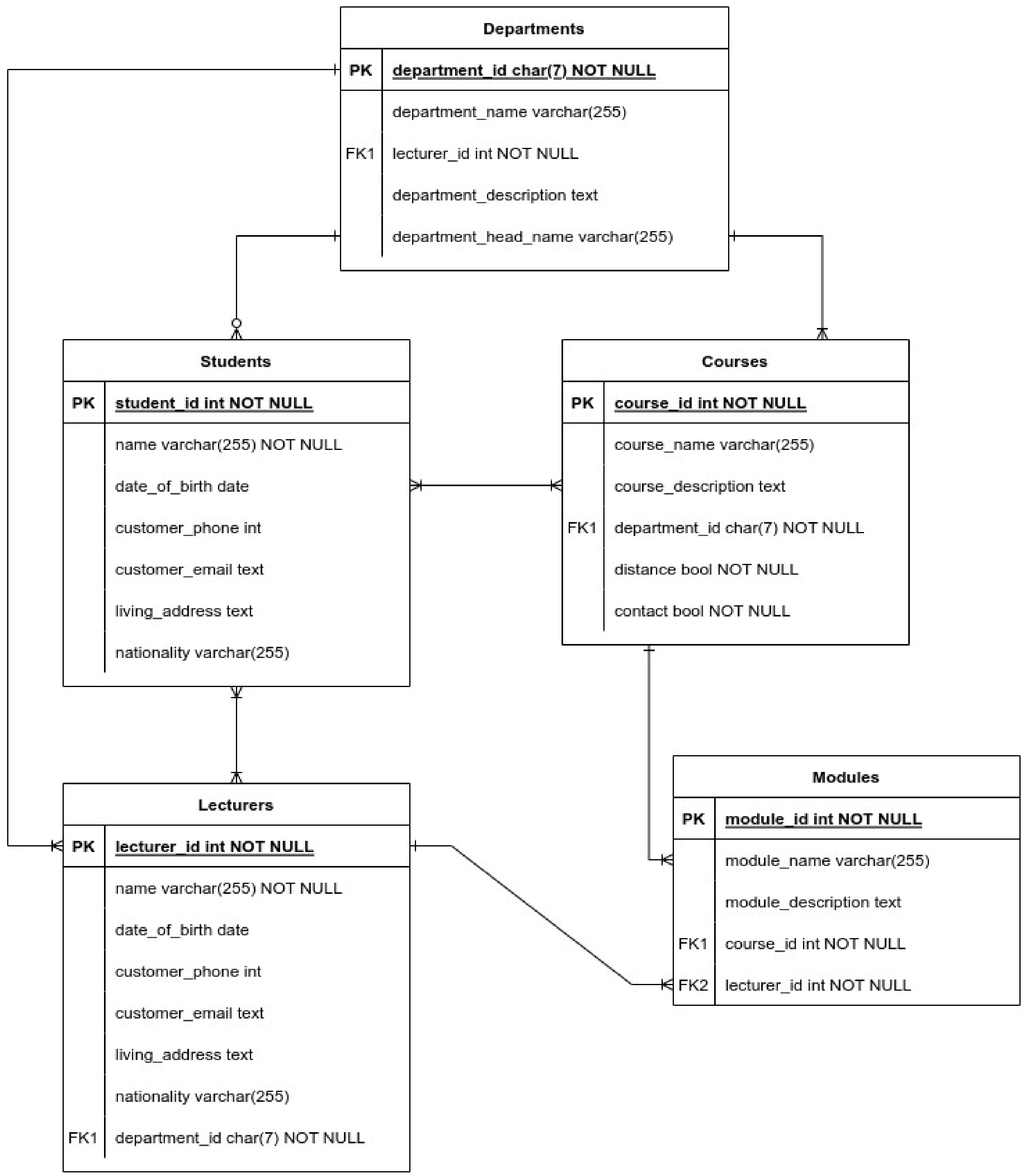
- → **Student:** *Attributes:* Student ID (Primary Key), Name, Age, Gender, Address, Email, Phone Number, Date of Birth, Graduation Date
- → **Teacher:** *Attributes:* Teacher ID (Primary Key), Name, Age, Gender, Address, Email, Phone Number, Department ID
- → **Courses:** *Attributes:* Course ID (Primary Key), Course Name, Department_ID, Level, Credits
- → **Modules:** *Attributes:* Module ID (Primary Key), Module Name, Course ID, Teacher ID, Description, Start Date, End Date

→ **Departments:** *Attributes:* Department ID (Primary Key), Department Name, Location, Head of Department

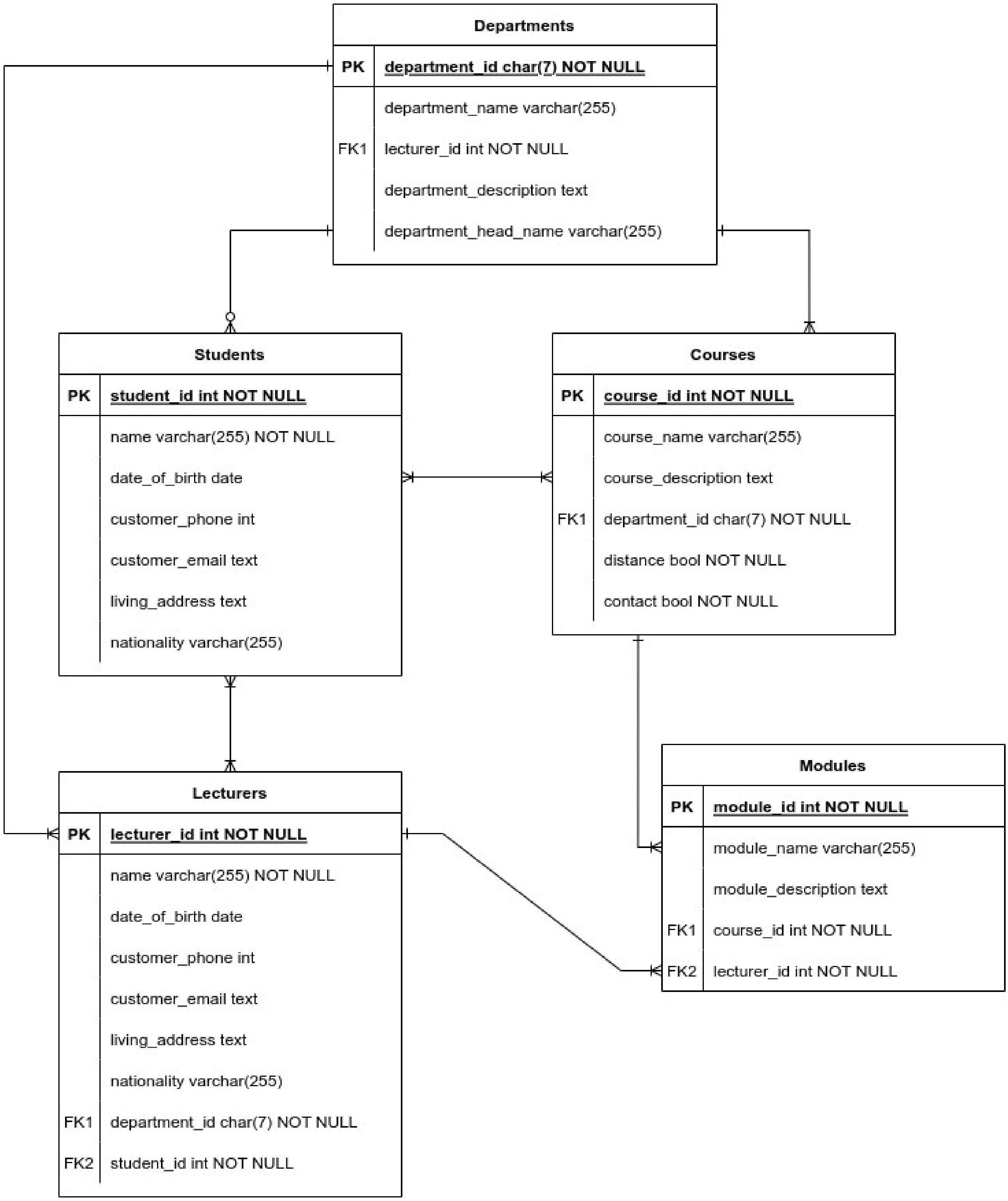
Question 3.3



Question 3.4


Departments
PK department_id char(7) NOT NULL
department_name varchar(255)
FK1 lecturer_id int NOT NULL
department_description text
department_head_name varchar(255)
Students
PK student_id int NOT NULL
name varchar(255) NOT NULL
date_of_birth date
customer_phone int
customer_email text
living_address text
nationality varchar(255)
Courses
PK course_id int NOT NULL
course_name varchar(255)
course_description text
FK1 department_id char(7) NOT NULL
distance bool NOT NULL
contact bool NOT NULL
Modules
PK module_id int NOT NULL
module_name varchar(255)
module_description text
FK1 course_id int NOT NULL
FK2 lecturer_id int NOT NULL
Lecturers
PK lecturer_id int NOT NULL
name varchar(255) NOT NULL
date_of_birth date
customer_phone int
customer_email text
living_address text
nationality varchar(255)
FK1 department_id char(7) NOT NULL

Question 3.5


Departments
PK department_id char(7) NOT NULL
department_name varchar(255)
FK1 lecturer_id int NOT NULL
department_description text
department_head_name varchar(255)
Students
PK student_id int NOT NULL
name varchar(255) NOT NULL
date_of_birth date
customer_phone int
customer_email text
living_address text
nationality varchar(255)
Courses
PK course_id int NOT NULL
course_name varchar(255)
course_description text
FK1 department_id char(7) NOT NULL
distance bool NOT NULL
contact bool NOT NULL
Modules
PK module_id int NOT NULL
module_name varchar(255)
module_description text
FK1 course_id int NOT NULL
FK2 lecturer_id int NOT NULL
Lecturers
PK lecturer_id int NOT NULL
name varchar(255) NOT NULL
date_of_birth date
customer_phone int
customer_email text
living_address text
nationality varchar(255)
FK1 department_id char(7) NOT NULL
FK2 student_id int NOT NULL

QUESTION FOUR

Question 4.1

Deletion Anomalies:

Deletion anomalies happen when deleting data from a database leads to unintended loss of other relevant information.

Example: Let's say we have a database of students where each student's information is stored along with the courses they are enrolled in. If we delete a student's record from the database, it may also delete the information about the courses they were taking, even if there are other students still enrolled in those courses.

Deletion anomalies happen when deleting one piece of data causes the loss of other relevant data. In the example, deleting a student's record should not result in the loss of course information.

Deletion anomalies can make it difficult to maintain data consistency and can lead to the loss of important information.

Update Anomalies:

Update anomalies happen when updating data in a database leads to inconsistencies or errors in the database.

Example: Consider the same database of students and their courses. If a student changes their major, we need to update their major information in the database. However, if we don't update all occurrences of that student's major across different records, inconsistencies can show up. For example, their major may appear differently in different places.

Update anomalies happen when updating data causes inconsistencies within the database. In the example, failing to update all occurrences of the student's major can result in a discrepancy, making it challenging to rely on the accuracy of the data. Update anomalies can make it difficult to maintain data integrity and can lead to confusion or incorrect information.

Insertion Anomalies:

Insertion anomalies happen when adding new data to a database is problematic because of the structure or dependencies of the database.

Example: Let's say we have a database of books where each book is associated with the author's information. If we want to add a new author to the database, but they haven't written any books yet, the database may not allow us to add the author without a corresponding book entry. Thus, we cannot insert the author's information unless they have written at least one book.

Insertion anomalies happen when it is difficult or impossible to add new data to the database because of the database's design or dependencies. In the example, we encounter an insertion anomaly because the database requires an author to have at least one book before their information can be inserted.

Insertion anomalies can hinder the flexibility and usability of the database, making it challenging to add new data when necessary.

Question 4.2

1st Normal Form (1NF)

1. Eliminate repeating groupIdentify the primary key
2. Identify All Dependencies

Primary Key (composite key) - designation, emp_no, dept_no

Employees Table - Attributes:

- ➞ Designation
- ➞ Emp_No
- ➞ Emp_Name
- ➞ Emp_Add
- ➞ Dept_No
- ➞ Dept_Name
- ➞ Dept_Location

Dependencies:

emp_no -> emp_name, emp_add

dept_no -> dept_name -> dept_location

dept_no -> designation

2nd Normal Form (2NF)

1. Make new tables to eliminate partial dependencies
2. Reassign matching dependent attributes

The relations meet the criteria for 2nd-normal form. There are no partial dependencies

Employees Table - Attributes:

- ➞ Emp_No (PK)
- ➞ Emp_Name

- ➞ Emp_Add

Department Table - Attributes:

- ➞ Dept_No (PK)
- ➞ Dept_Name
- ➞ Dept_Location
- ➞ Designation

3rd Normal Form (3NF)

1. Make new tables to eliminate transitive dependencies
2. Reassign matching dependent attributes

The relations meet the criteria for 3rd-normal form, there are no transitive dependencies

Employees Table - Attributes:

- ➞ Emp_No (PK)
- ➞ Emp_Name
- ➞ Emp_Add

Department Table - Attributes:

- ➞ Dept_No (PK)
- ➞ Dept_Name
- ➞ Dept_Location
- ➞ Designation

Boyce Codd Normal Form (BCNF)

1. All keys must be candidate keys

Does not meet the BCNF as every key is not a candidate key

Employees Table - Attributes:

- ➞ Emp_No (PK)
- ➞ Emp_Name
- ➞ Emp_Add

Department Table - Attributes:

- ➞ Dept_No (PK)
- ➞ Dept_Name
- ➞ Dept_Location
- ➞ Designation

Question 4.3

The Dewey Decimal System works by breaking books into different categories with numeric.
Question 4.4

A relational database organizes data by breaking it up into attributes. The Dewey Decimal system uses a system of categorization with lends itself to the paradigm of an RDBS.

A RDBS can store each level of the Dewey Decimal system as a different table.

There are ten main subject areas: The system is divided into ten main subject areas, each represented by a specific numerical range:

000-099: General Knowledge and Information

100-199: Philosophy and Psychology

200-299: Religion

300-399: Social Sciences

400-499: Language

500-599: Natural Sciences and Mathematics

600-699: Technology and Applied Sciences

700-799: Arts and Recreation

800-899: Literature

900-999: History, Geography, and Biography

Question 4.5

A RDBS expects to deal with structured data.

Semi-Strutcured or Unstructed data is unsuited to being managed by a relational database. A RDBS requires the definer to know before hand what tables to create and how they relate. This makes a RDBS inflexible.

References:

- Carlos Coronel and Steven Morris (2019) *Database Systems: Design, Implementation, and Management, 13th Edition.* Cengage Learning
- Entity Relationship Diagram (ERD) Tutorial - Part 1, 2017. Lucid Softwrae. [Online] Available at: https://www.youtube.com/watch?v=QpdhBUYk7Kk
  [Accessed 20 May 2023].