Database
Management Systems

O.B.E 2.0

Add a short description here

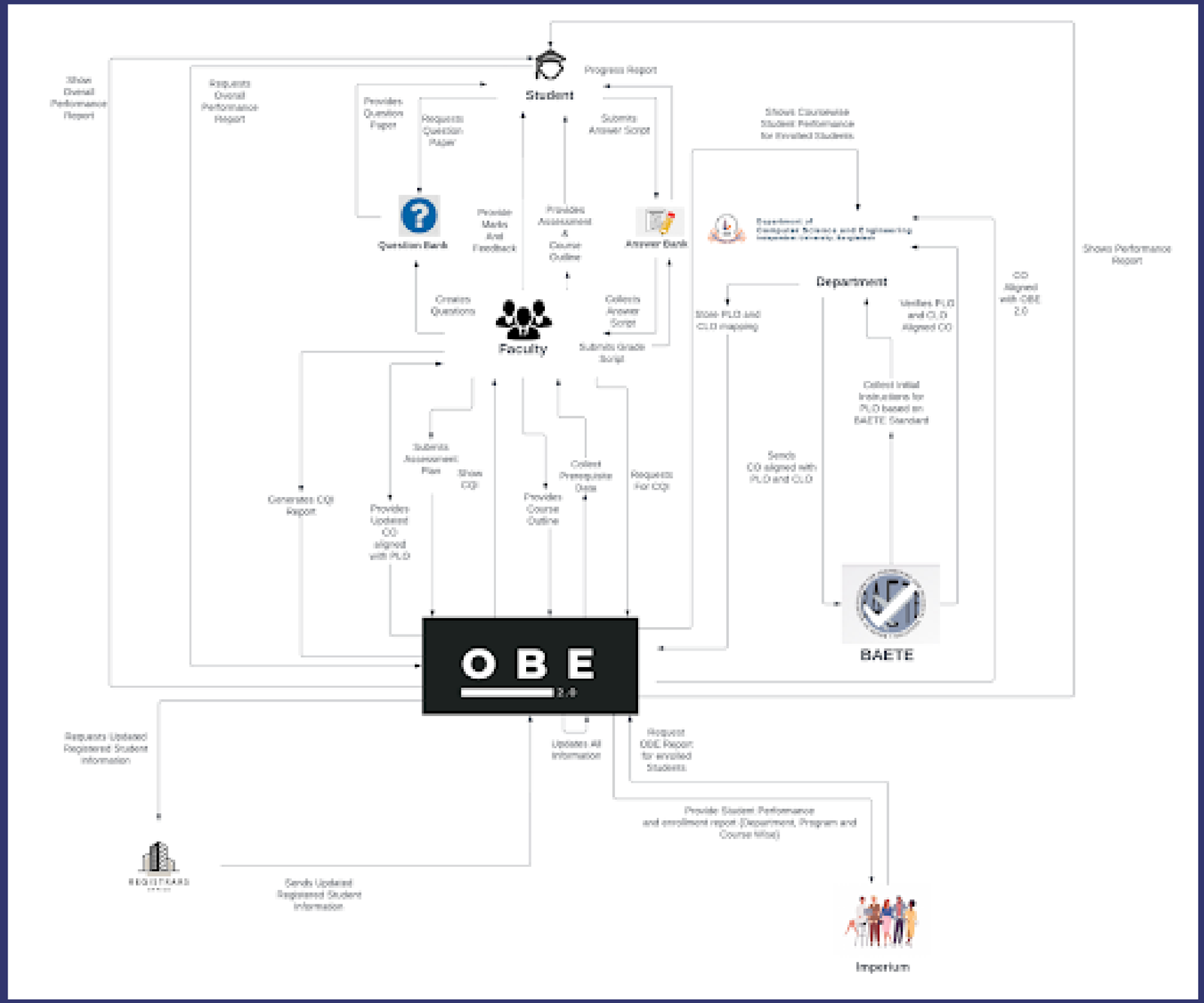
Text-to-speech script available.

Objective

- The main objective of our project is to implement a user-friendly monitoring software that will improve the quality of education by allowing institutional bodies and students to track progress more productively. By visualizing progress, courses can be made more effective and students can clearly see not only their progress but also which skills a course is helping them achieve. Our hope for the project is to help advance all curriculums, not only CSE, as well as the students as an individual.

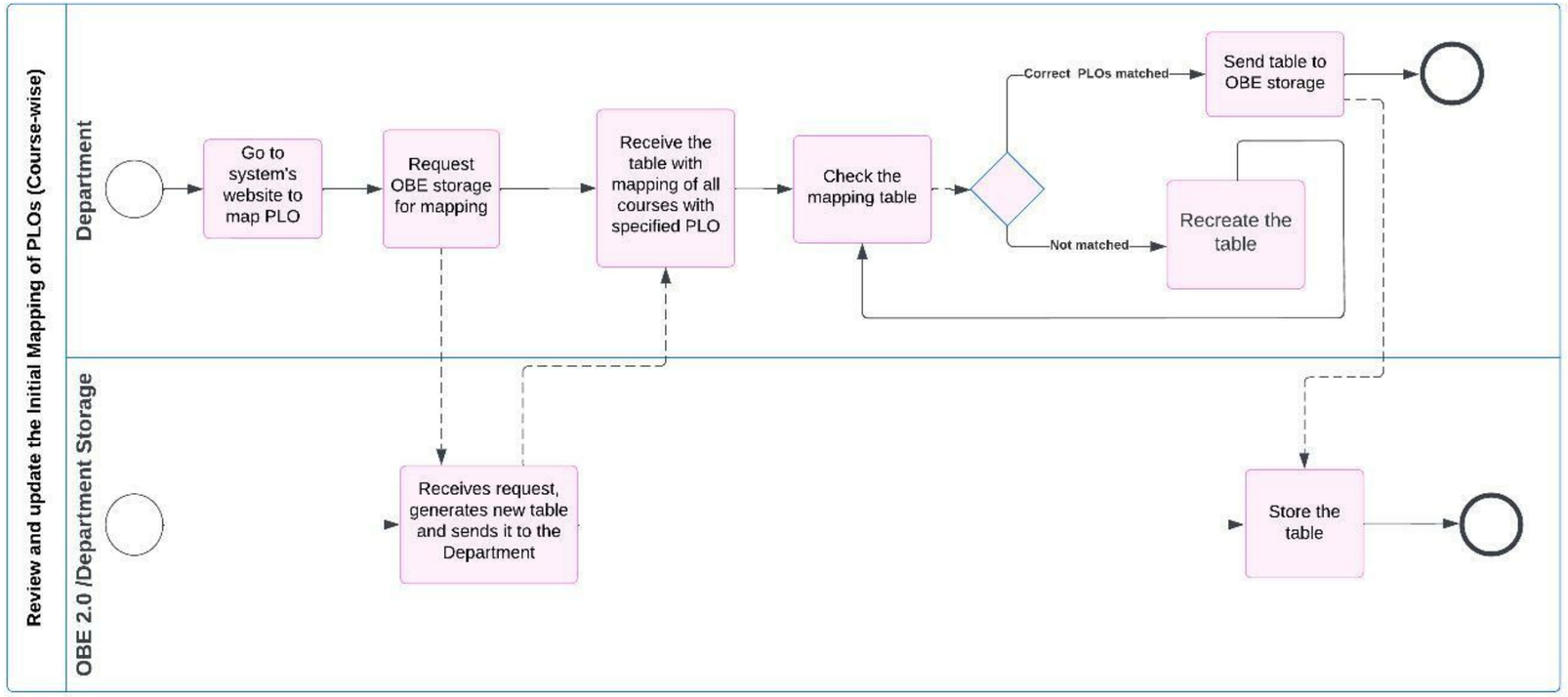
Rich Picture

A Rich Picture is a way to explore, acknowledge and define a situation and express it through diagrams to create a preliminary mental model.



Database
Management Systems

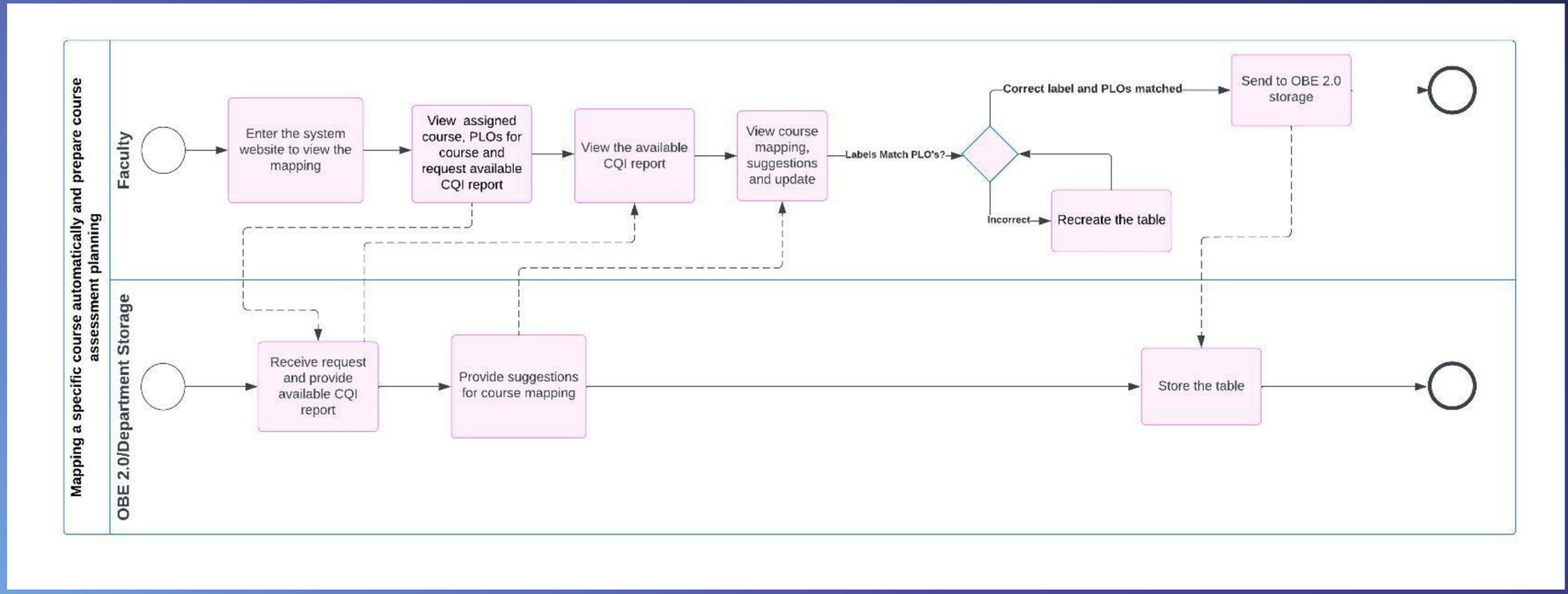
Process Diagram (BPMN)



Back to Agenda Page

Database
Management Systems

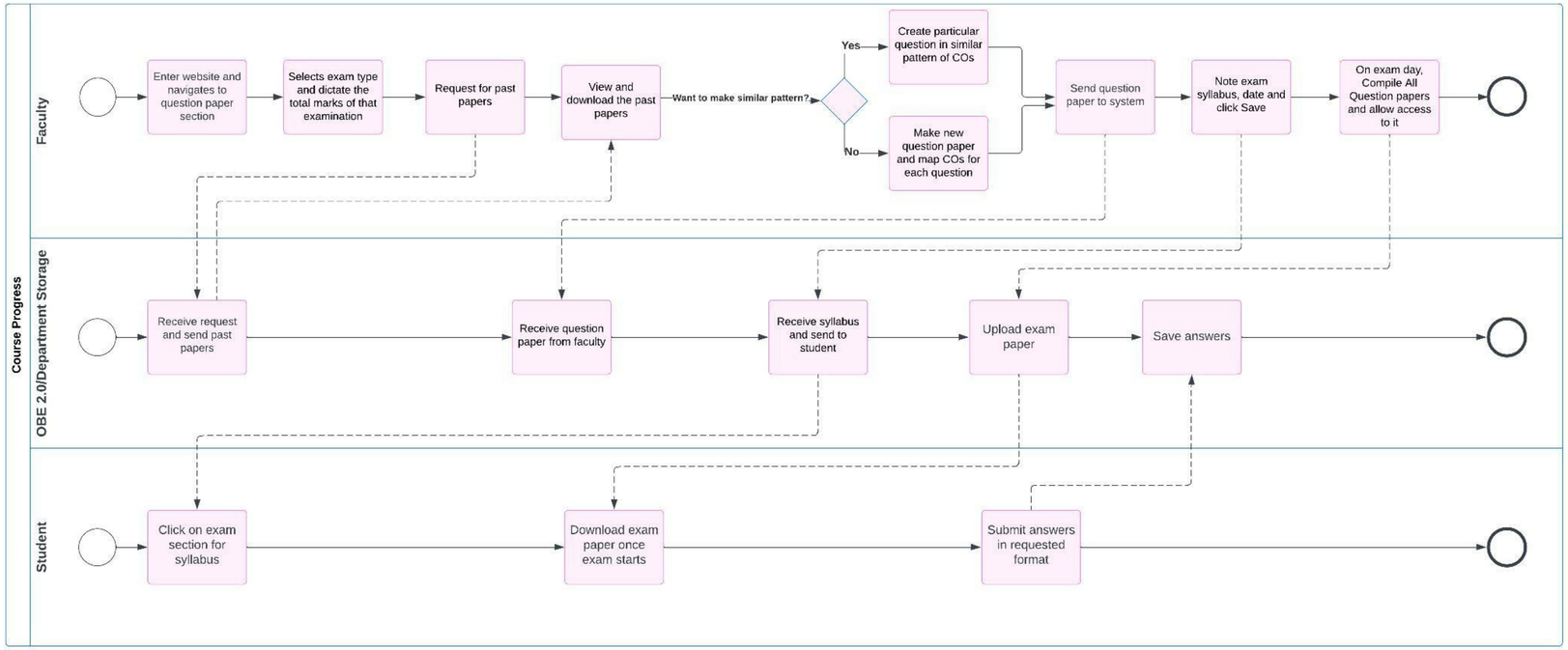
Process Diagram (BPMN)



Back to Agenda Page

Database
Management Systems

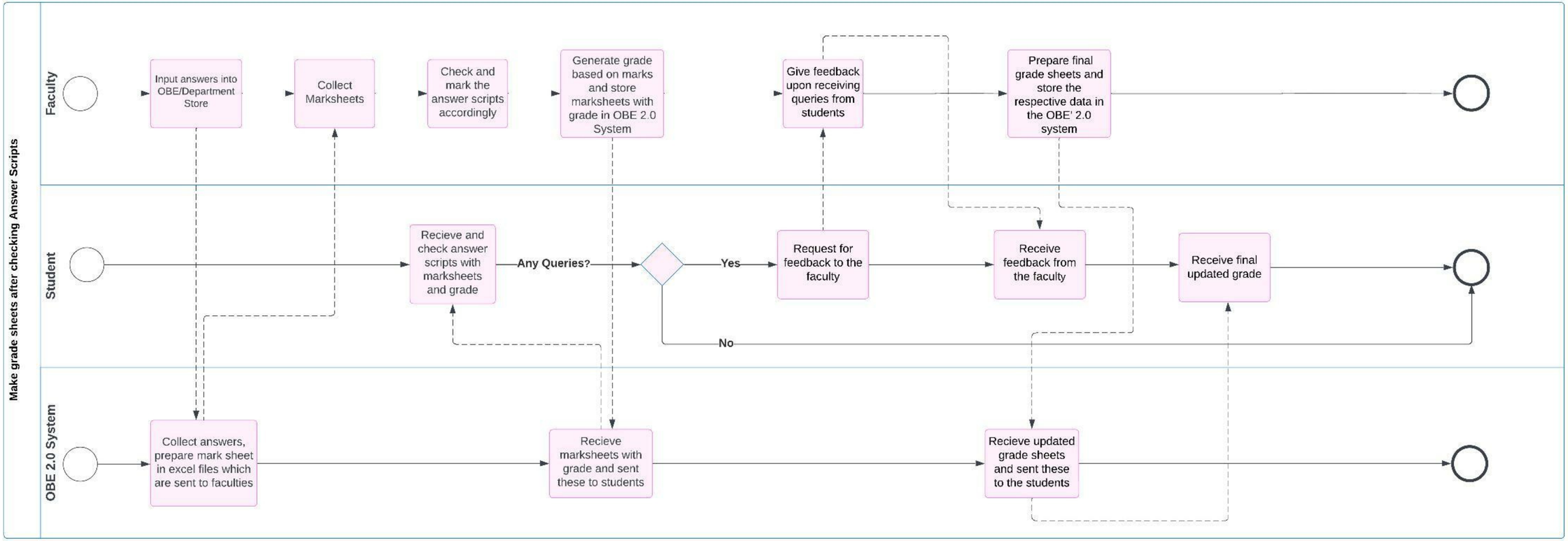
Process Diagram (BPMN)



Back to Agenda Page

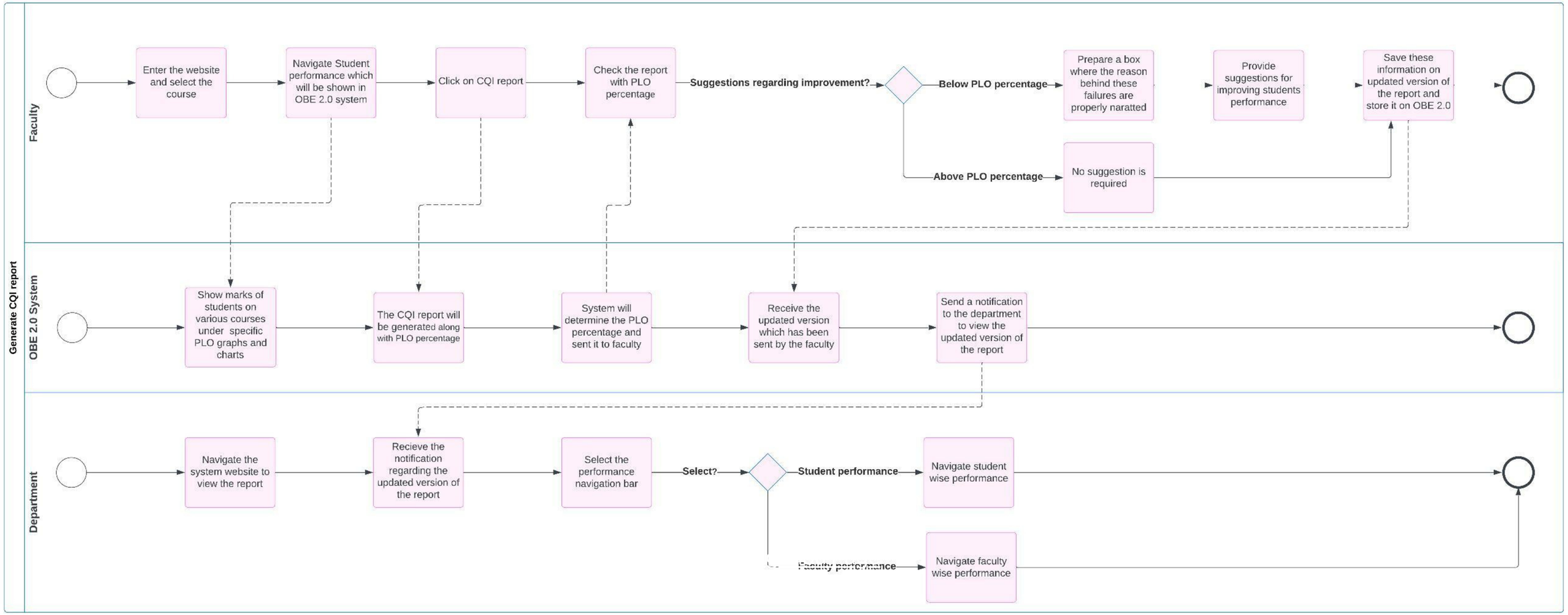
Database
Management Systems

Process Diagram (BPMN)



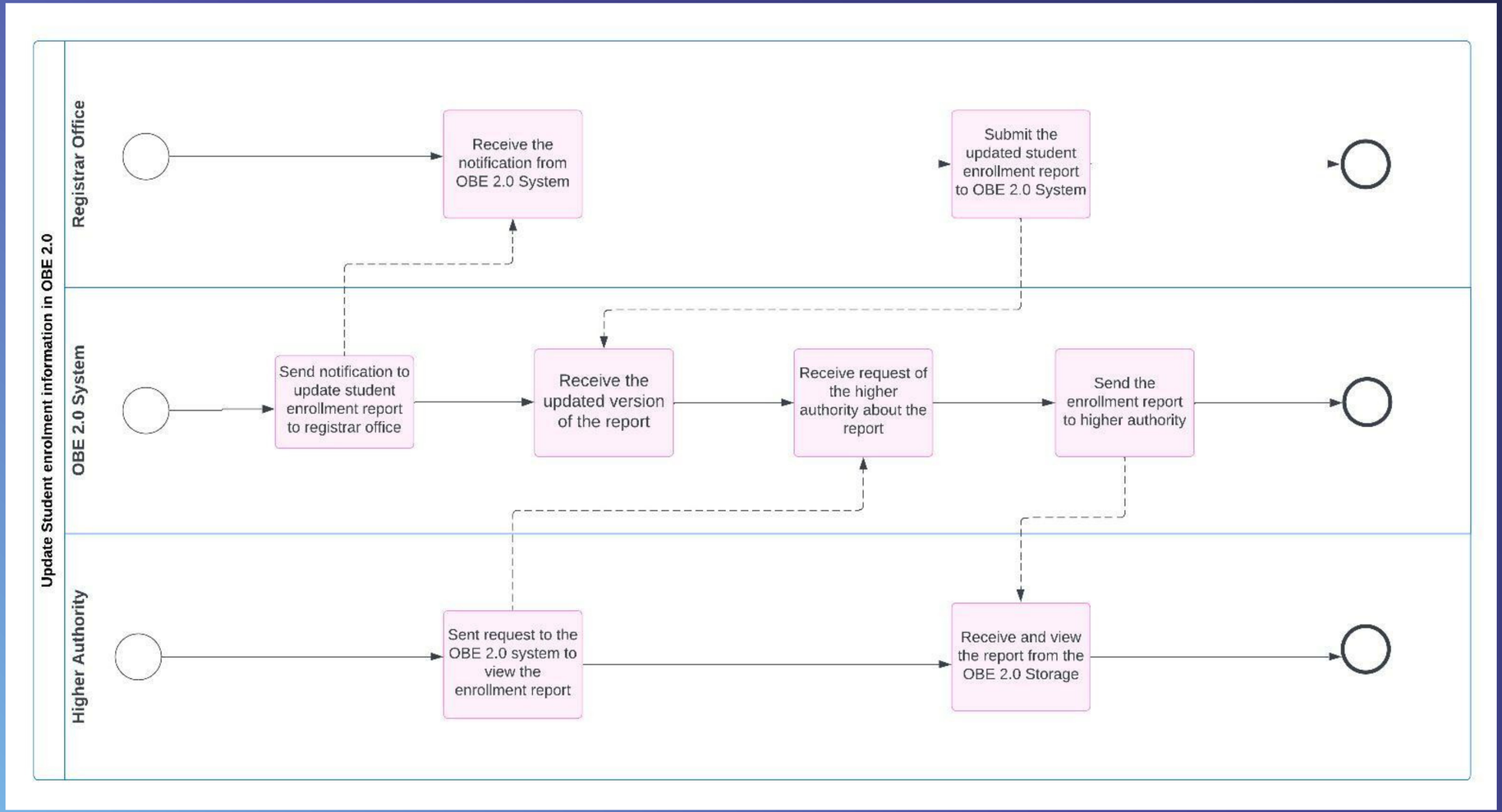
Database
Management Systems

Process Diagram (BPMN)



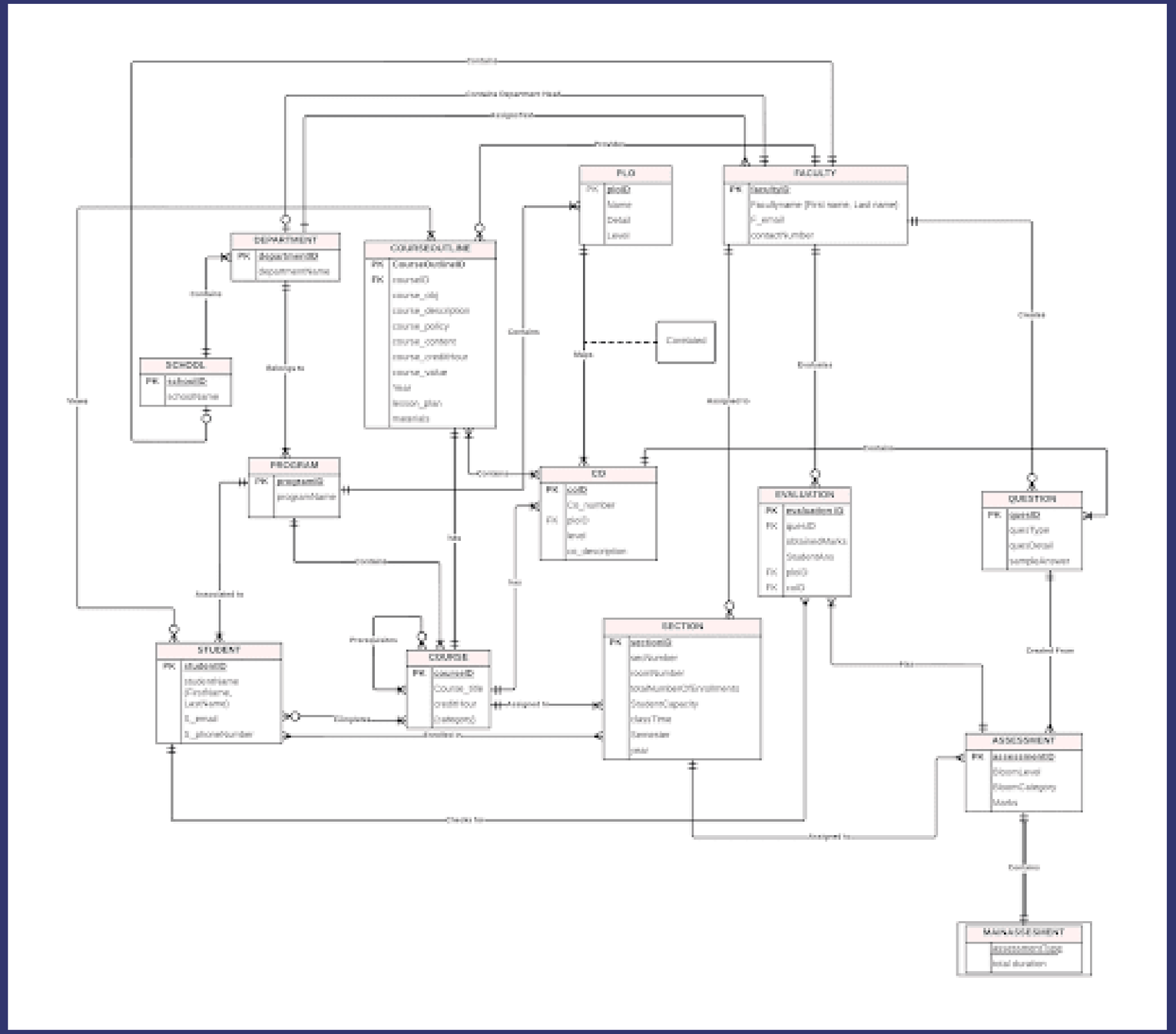
Database
Management Systems

Process Diagram (BPMN)

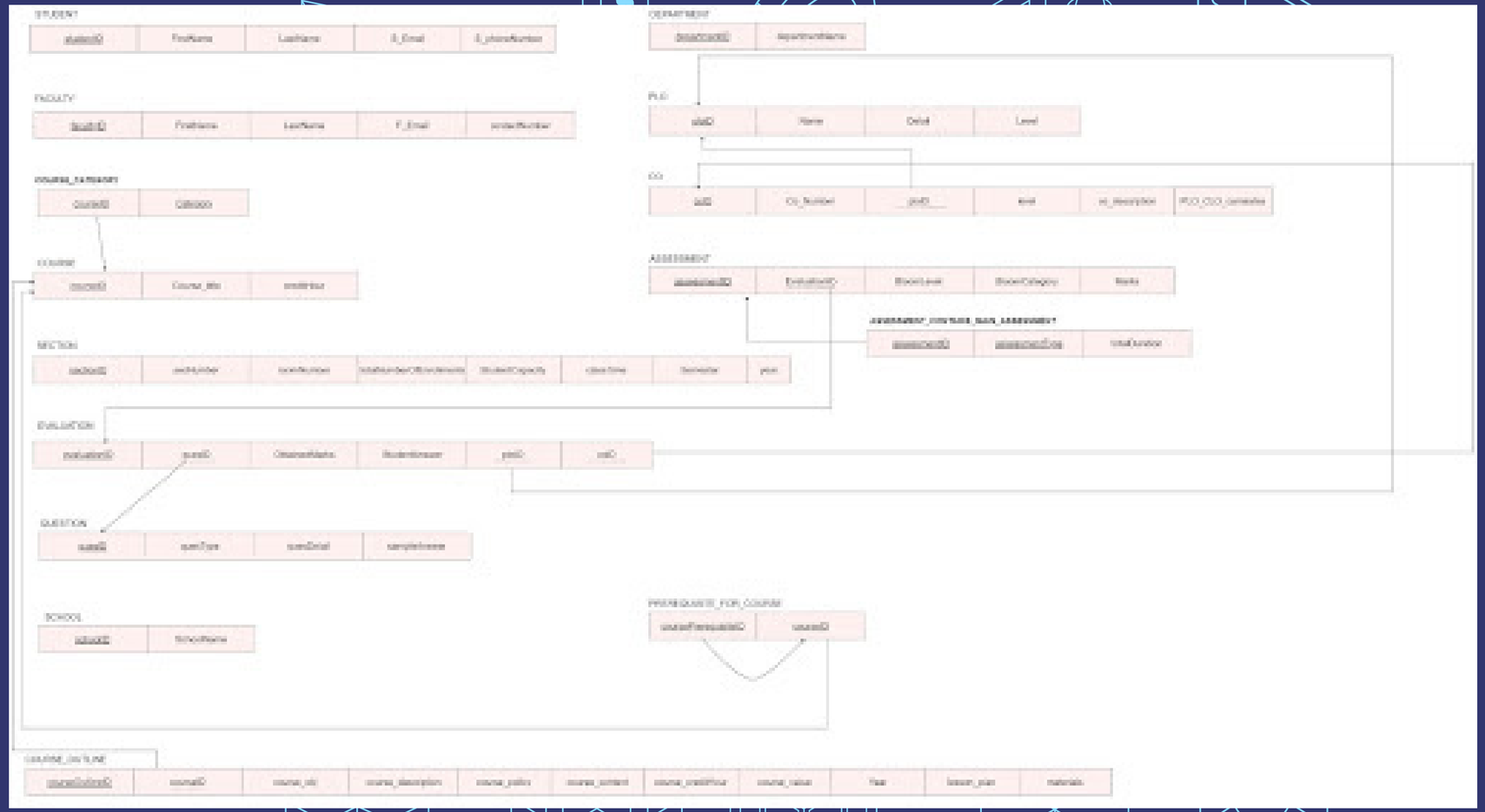


ERD

An entity–relationship Diagram describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.



Relational Schema



A relational schema is a set of relational tables and associated items that are related to one another

Normalization Indexing

Normalization is a process of decomposing relations with animalities to produce smaller, well structured relations. The normalization of our project was a bit tricky, so we indexed every attribute of our entities with letters. Then, we have normalized the problem in 1NF, 2NF, 3NF and in BCNF.

t1→	t2, t3, t4, t5
d1→	d2
f1→	f2, f3, f4, f5
p1→	p2, p3, p4
c1→	c2, p1, c3, c4, c5
o1→	o2, o3, v1, u1, w1
a1→	e1, a2, a3, a4, r1, r2
b1→	b2, b3, b4, b5, b6, b7, b8
e1→	q1, e2, e3, p1, c1
q1→	q2, q3, q4
s1→	s2
w1→	w2, w3, w4, w5, w6, w7, w8, w9, w10

studentID→	FirstName, LastName, S_email, S_phoneNumber
departmentID→	departmentName
facultyID→	FirstName, LastName, F_email ,contactNumber
coID→	Co_Number, ploID, level, co_description, PLO_CLO correlates
courseID→	Course_Title, creditHour, Category, coursePrerequisiteID, courseOutlineID
assesmentID→	evaluationID, BloomLevel, BloomCategory, Marks, assesmentType, totalDuration
sectionID→	secNumber, roomNumber, totalNumberofEnrollments, StudentCapacity, classTime, Semester, year
evaluationID→	quesID, ObtainedMarks, StudentAnswers, ploID, coID
quesID→	quesType, quesDetail, sampleAnswer
schoolID→	SchoolName
courseOutlineID→	course_obj, course_description, course_policy, course_content, course_creditHour, course_value, Year, lesson_plan, materials

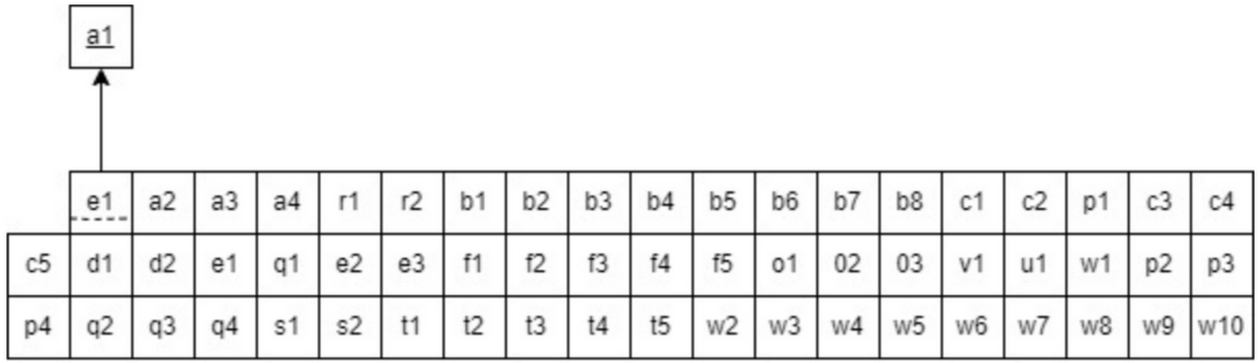
Normalization

In 1nf, we have determined the primary key which is **a1** since it uniquely identifies majority of the non-key attributes. In 2nf, we have made every non-key attribute fuly functionally dependent on primary key **a1 .**

1nF

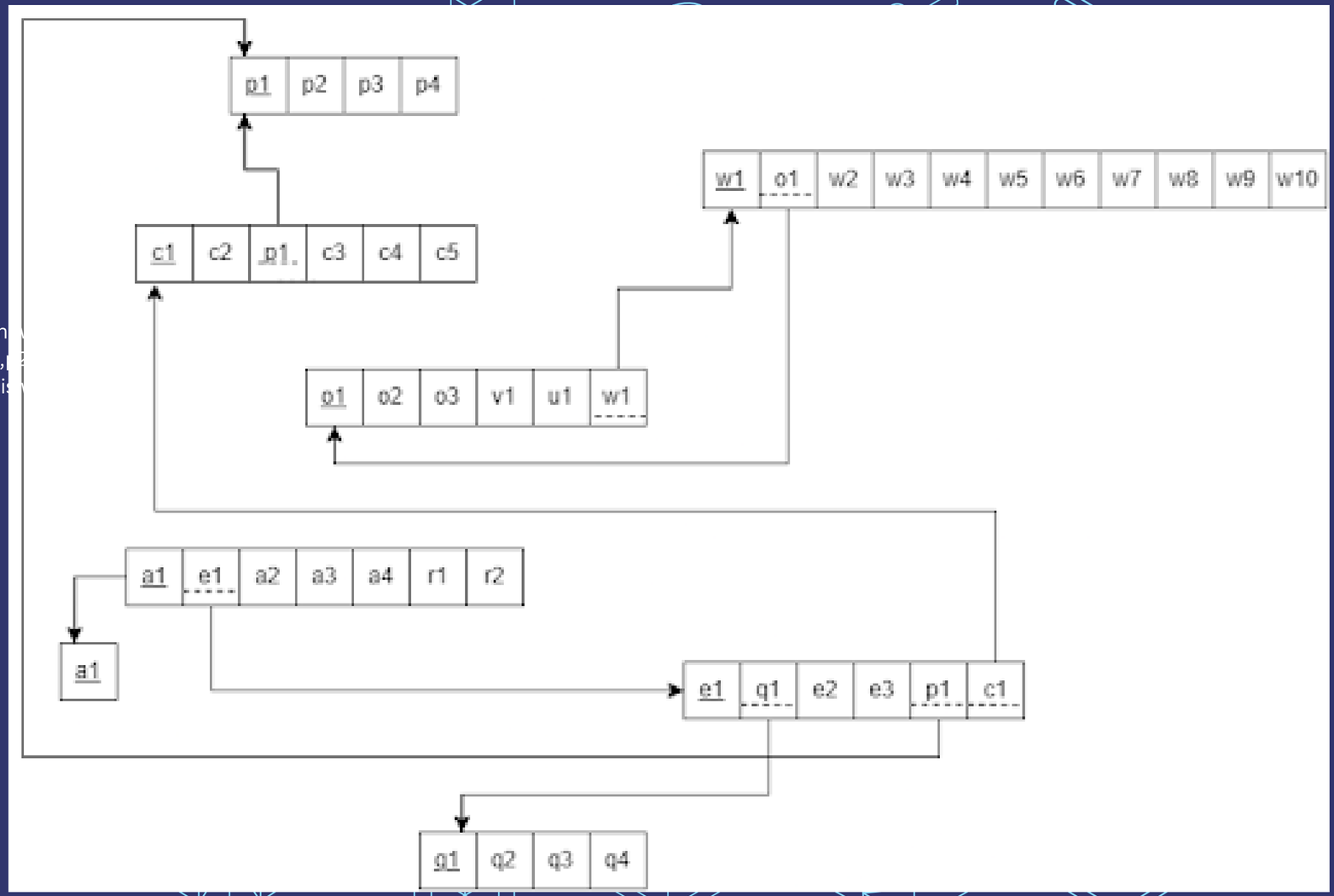
a1	e1	a2	a3	a4	r1	r2	b1	b2	b3	b4	b5	b6	b7	b8	c1	c2	p1	c3	c4
c5	d1	d2	e1	q1	e2	e3	f1	f2	f3	f4	f5	o1	02	03	v1	u1	w1	p2	p3
p4	q2	q3	q4	s1	s2	t1	t2	t3	t4	t5	w2	w3	w4	w5	w6	w7	w8	w9	w10

2nF



Normalization 3NF

In 3NF, we removed transitive dependency for each non-key attributes by creating n
relations. For example- p1 is a non key attribute which is primary key in the relation p1,
and p4. It's also a foreign key in the c1 primary key table and e1 primary key table. In this
transitive dependancy is removed.



BCNF

- In this stage of our normalization, no non-key attribute can identify any primary key or part of the primary key. So, we can say that all relations are already in BCNF.

SQL Queries

SELECT * FROM question;

This will allow the faculty to access details in the table.

INSERT INTO question(quesID, quesType, quesDetail, sampleAns)
VALUES ('1','easy','Write the smallest prime number in range 0-10','3');

The system will allow the faculty to add new questions to the question bank. The values entered are all examples.

SELECT * FROM assessment;

This will allow the faculty to access details in the table.

INSERT INTO assessment(assessmentID,bloomLevel, bloomCategory,marks)
VALUES ('1','1','remembering','10');

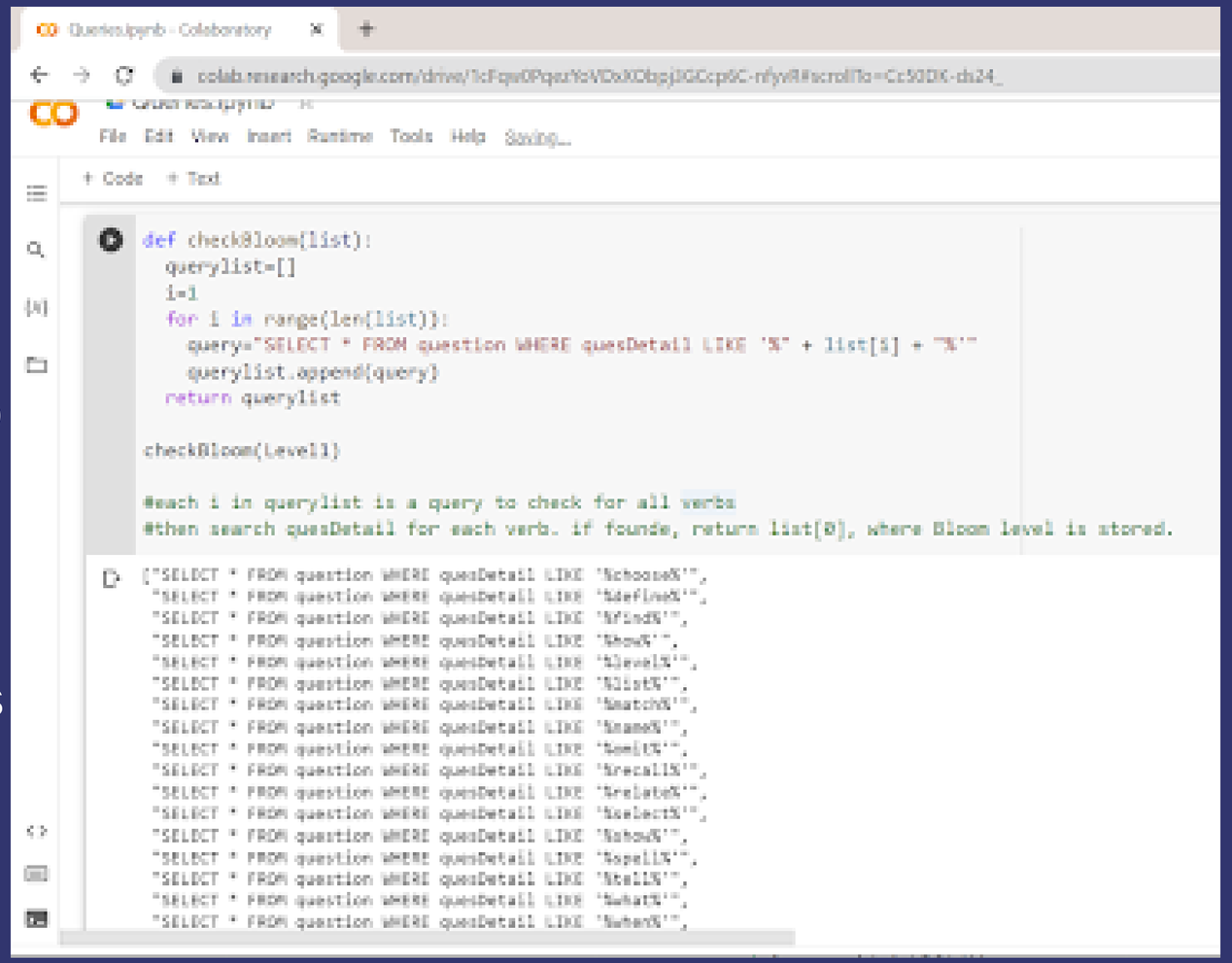
The system will allow the faculty to add assessment for questions. The values entered are all examples.

SQL Queries

**SELECT *
FROM question
WHERE quesDetail LIKE '%verb%';**

This will search the quesDetail column which contains questions to search for a specific verb given and fetch the rows in question where the verb appears.

To make it easier to check for every verb, the following python code was made to create verb lists for every level, where the first index contains the Bloom level information:



SQL Queries

SELECT * FROM courseoutline;

This will allow the faculty to access details in the table.

INSERT INTO courseoutline(courseOutlineID,courseID,courseObj,courseDescription,coursePolicy,courseContent,creditHour,courseValue,currentYear,lessonPlan,materials)

VALUES (‘1’,'CSE303','Learn DBMS','DBMS','No plagiarism','SQL','4','high','2022','plan','books');

The system will allow faculty to add new course outlines to the course outline table in the database. The values entered are all examples.

INSERT INTO plo(ploID,ploLevel,ploName,ploDetails) VALUES ('1','1','1stplo','details');

Allows faculty users to insert PLO details into the database table. The values entered are all examples.

INSERT INTO co(coID,coLevel,coNumber,coDescription,ploID) VALUES ('1','1','1','desc','1');

Allows faculty users to insert CO details into the database table. The values entered are all examples.