

LMS Database

Presented by Ali Khantszian
Professor: Esenalieva Gulzat

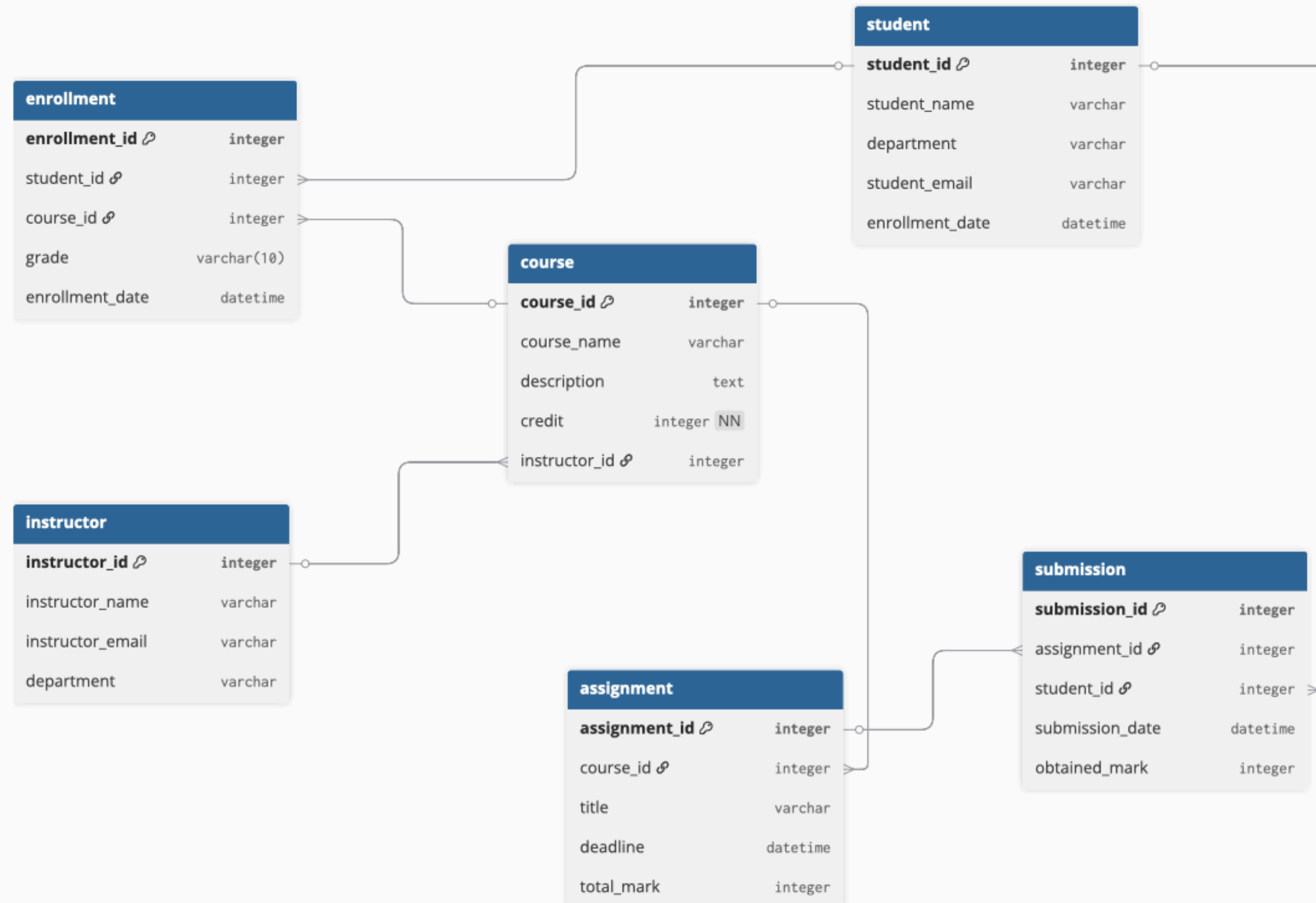
Table of content

- Introduction
- ER-diagram
- Scheme
- Basic queries
- Advanced queries
- Transactions
- Indexing
- Backup and Restore
- Conclusion

Introduction

A Learning Management System (LMS) is a platform used by educational institutions to organize courses, manage students, track academic progress, and streamline communication between teachers and learners. The goal of this project is to design and implement a complete relational database for an LMS.

ER-diagram



Scheme

```
1  CREATE DATABASE lms_db;
2  \c lms_db;
3
4
5  CREATE TABLE student (
6      student_id SERIAL PRIMARY KEY,
7      student_name VARCHAR(100) NOT NULL,
8      department VARCHAR(100),
9      student_email VARCHAR(150) UNIQUE NOT NULL,
10     enrollment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
11 );
12
13
14 CREATE TABLE instructor (
15     instructor_id SERIAL PRIMARY KEY,
16     instructor_name VARCHAR(100) NOT NULL,
17     instructor_email VARCHAR(150) UNIQUE NOT NULL,
18     department VARCHAR(100)
19 );
20
21
22 CREATE TABLE course (
23     course_id SERIAL PRIMARY KEY,
24     course_name VARCHAR(150) NOT NULL,
25     description TEXT,
26     credit INTEGER NOT NULL CHECK (credit > 0),
27     instructor_id INTEGER REFERENCES instructor(instructor_id)
28         ON DELETE SET NULL
29 );
30
```

Scheme

```
32 CREATE TABLE enrollment (  
33     enrollment_id SERIAL PRIMARY KEY,  
34     student_id INTEGER REFERENCES student(student_id)  
35         ON DELETE CASCADE,  
36     course_id INTEGER REFERENCES course(course_id)  
37         ON DELETE CASCADE,  
38     grade VARCHAR(10),  
39     enrollment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
40 );  
41  
42  
43 CREATE TABLE assignment (  
44     assignment_id SERIAL PRIMARY KEY,  
45     course_id INTEGER REFERENCES course(course_id)  
46         ON DELETE CASCADE,  
47     title VARCHAR(150) NOT NULL,  
48     deadline TIMESTAMP NOT NULL,  
49     total_mark INTEGER NOT NULL CHECK (total_mark > 0)  
50 );  
51  
52  
53 CREATE TABLE submission (  
54     submission_id SERIAL PRIMARY KEY,  
55     assignment_id INTEGER REFERENCES assignment(assignment_id)  
56         ON DELETE CASCADE,  
57     student_id INTEGER REFERENCES student(student_id)  
58         ON DELETE CASCADE,  
59     submission_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
60     obtained_mark INTEGER CHECK (obtained_mark >= 0)  
61 );
```

Basic queries

```
1  SELECT * FROM student;
2
3  SELECT * FROM course;
4
5  SELECT s.student_name, c.course_name, e.grade
6  FROM enrollment e
7  JOIN student s ON e.student_id = s.student_id
8  JOIN course c ON e.course_id = c.course_id;
```

Output of queries

student_id	student_name	department	student_email	enrollment_date
1	Ali Khatszian	Software Engineering	ka12438@auca.kg	2025-12-12 16:55:29.550158
2	Ikbol Allakhunov	Software Engineering	ai12351@auca.kg	2025-12-12 16:55:29.550158
3	Anne Wallace	Business Administration	Anne.Wallace@gmail.com	2025-12-12 16:55:29.550158
4	Lauren Clark	LAS	Lauren.Clark@soprasteria.com	2025-12-12 16:55:29.550158
5	Christopher Duffy	AMI	Christopher.Duffy@yahoo.com	2025-12-12 16:55:29.550158
6	Matthew Landry	Economic	Matthew.Landry@gmail.com	2025-12-12 16:55:29.550158

(6 rows)

course_id	course_name	description	credit	instructor_id
1	Database	Introduction to databases	6	1
2	Macroeconomic	Intrduction to Macroeconomic theory	6	2
3	Financial accounting	Financial accounting 1	6	3
4	Social Entrepreneurship	Intrduction to Social Entrepreneurship	6	4
5	Philosophy of law	Introduction to law theory and philosophy of law	6	5
6	Data Structures	Intro to Data Structures	6	1

(6 rows)

student_name	course_name	grade
Ikbol Allakhunov	Database	A-
Anne Wallace	Macroeconomic	B
Lauren Clark	Philosophy of law	C
Christopher Duffy	Macroeconomic	X
Lauren Clark	Social Entrepreneurship	B+
Anne Wallace	Financial accounting	B
Lauren Clark	Social Entrepreneurship	B-
Ali Khatszian	Financial accounting	N/A
Ali Khatszian	Database	A

(9 rows)

Advanced queries

```
1  SELECT s.student_name, AVG(sub.obtained_mark) AS average_mark
2  FROM submission sub
3  JOIN student s ON sub.student_id = s.student_id
4  GROUP BY s.student_name
5  ORDER BY average_mark DESC;
6
7  SELECT c.course_name, COUNT(e.student_id) AS total_students
8  FROM course c
9  LEFT JOIN enrollment e ON c.course_id = e.course_id
10 GROUP BY c.course_name;
11
12 SELECT s.student_name, a.title, sub.submission_date, a.deadline
13 FROM submission sub
14 JOIN assignment a ON sub.assignment_id = a.assignment_id
15 JOIN student s ON sub.student_id = s.student_id
16 WHERE sub.submission_date > a.deadline;
```

Output of queries

student_name	average_mark
Ali Khatszian	96.5000000000000000
Ikbol Allakhunov	81.0000000000000000
Lauren Clark	80.0000000000000000

(3 rows)

course_name	total_students
Data Structures	0
Philosophy of law	1
Financial accounting	2
Social Entrepreneurship	2
Macroeconomic	2
Database	2

(6 rows)

student_name	title	submission_date	deadline
Ikbol Allakhunov	ER Diagram Project	2025-12-12 16:55:29.576639	2025-12-01 23:59:00
Ali Khatszian	ER Diagram Project	2025-12-12 16:55:29.576639	2025-12-01 23:59:00
Lauren Clark	Homework	2025-12-12 16:55:29.576639	2025-12-05 23:59:00
Ikbol Allakhunov	Group assignment	2025-12-12 16:55:29.576639	2025-12-05 23:59:00

(4 rows)

Transactions

```
1  -- Enrolling a Student
2  BEGIN;
3
4  -- Verify that the student exists
5  SELECT * FROM student WHERE student_id = 1 FOR SHARE;
6
7  -- Verify the course exists
8  SELECT * FROM course WHERE course_id = 3 FOR SHARE;
9
10 -- Insert enrollment
11 INSERT INTO enrollment (student_id, course_id, grade)
12 VALUES (1, 3, 'N/A');
13
14 COMMIT;
15
16 -- Student Submits
17 BEGIN;
18
19 -- Add submission
20 INSERT INTO submission (assignment_id, student_id, obtained_mark)
21 VALUES (6, 1, 98)
22 RETURNING submission_id;
23
24 -- Update enrollment grade
25 UPDATE enrollment
26 SET grade = 'A'
27 WHERE student_id = 1 AND course_id = 1;
28
29 COMMIT;
30
31 -- Instructor Creates a New Course && assignment
32 BEGIN;
33
34 -- Create the course
35 INSERT INTO course (course_name, description, credit, instructor_id)
36 VALUES ('Data Structures', 'Intro to Data Structures', 6, 1)
37 RETURNING course_id;
38
39 -- Create assignment for the course
40 INSERT INTO assignment (course_id, title, deadline, total_mark)
41 VALUES (currval('course_course_id_seq'), 'First Homework', '2025-12-10 23:59:00', 100);
42
43 COMMIT;
```

Output of Transactions

student_id	student_name	department	student_email	enrollment_date
1	Ali Khatszian	Software Engineering	ka12438@auca.kg	2025-12-12 16:55:29.550158

(1 row)

course_id	course_name	description	credit	instructor_id
3	Financial accounting	Financial accounting 1	6	3

(1 row)

```
INSERT 0 1
COMMIT
BEGIN
  submission_id
```

6

(1 row)

```
INSERT 0 1
UPDATE 1
COMMIT
BEGIN
  course_id
```

7

(1 row)

```
INSERT 0 1
INSERT 0 1
COMMIT
```

Indexing

```
1 CREATE INDEX idx_student_email ON student(student_email);  
2 CREATE INDEX idx_course_name ON course(course_name);  
3 CREATE INDEX idx_enrollment_student ON enrollment(student_id);  
4 CREATE INDEX idx_submission_student ON submission(student_id);
```

Backup and restore

1. Backup Database:

```
pg_dump -U username -d lms_db -F c -f lms_backup.dump
```

2. Restore Database:

```
pg_restore -U username -d lms_db -c lms_backup.dump
```

Conclusion

In this project, a basic LMS database was designed and implemented, including basic and advanced queries, transactions, indexing, and a backup-and-restore strategy. where instructors could organize courses, manage students, track academic progress, and streamline communication with learners.

Thank You

made by Ali Khantszian