Project 1: Course Management System

Abhinav Jain (2019CS10322), Arnav Tuli (2019CS10424), Prabhakar Chaudhary (2019CS10381)

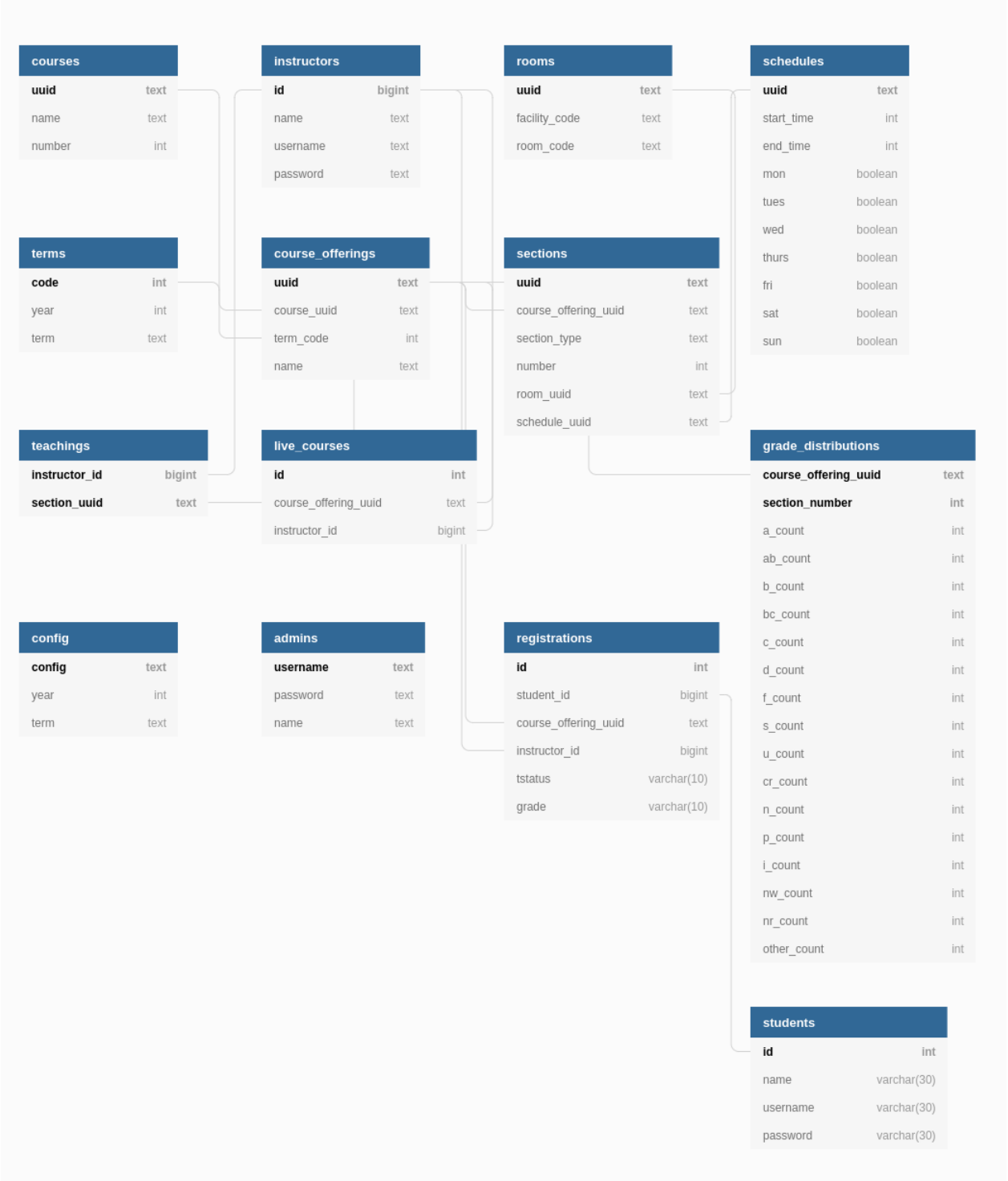
Due date: March 2, 2022, 9:00am IST

1 Section 1

Motivation: We decided to build a CMS, so as to present the University data to the users in a friendly manner, and provide fast-paced and dynamic interaction between students and instructors, with admin as the controller. The CMS allows students to register for a course, audit/withdraw a course during a specific window (set by admin), and view his/her grades after the instructor finishes uploading the grades. Students can also look at courses taught in the past and various statistics involving them, like grade distributions, timings, instructors etc., and can therefore make a more informed decision while registering for a course. Instructors can float courses (again within a window), Accept/Reject student requests, and grade students. Admin has the ability to change system configuration, which include: current window (registration, audit/withdrawal, grades etc.), term and year. Admin can also add/delete users and view users to get an overview of the system.

Complete System: We took inspiration from IITD e-Acads portal, and the complete system should be something like it, with different options available to different users based on their privileges (stud/ins/adm)

ER-diagram: PTO



(a)

Figure 1: ER-Diagram

2 Section 2

Data Sources

1. Majority of data was ready-made and taken directly from: Kaggle
2. We deleted two tables (subjects and subjects_memberships), as we were not using them. Also, ee added a few tables of our own such as config, students, admins, registrations and live_courses.
3. Term codes for corresponding year and term, were scraped from here, using Gecko-Driver (scraper.py)

S.No.	Table Name	No. of tuples	Loading Time (in ms)	Size of Table
1	admins (synthetic)	3	0.46	32 kB
2	config (synthetic)	1	0.387	64 kB
3	course_offerings	81462	152.619	18 MB
4	courses	9316	7.221	2128 kB
5	grade_distributions	193262	380.415	43 MB
6	instructors	18737	10.801	4000 kB
7	live_courses (organic)	10	0.27	48 kB
8	registrations (organic)	30	0.226	48 kB
9	rooms	1350	1.857	264 kB
10	schedules	4477	4.324	728 kB
11	sections	315612	797.424	84 MB
14	students (synthetic)	5	0.224	48 kB
15	teachings	315221	291.146	52 MB
16	terms (scraped)	150	1.141	48 kB

Synthetic: Generated manually
Organic: Generated via user interaction

3 Section 3

3.1 User View

- There are three types of users: Students, Instructors and Admins. The UI is specialized to each user. All users have a 'Home' page that consists of an avatar, name, username and buttons to change password and log-out.
- **Student View:** Students also have access to the 'Registration' page. Here, students can register for active/live courses (add/remove). During the audit/withdrawal period, students can also audit/withdraw from a registered course. Note that a student can have only 8 active/pending courses at a time. However, if the instructor rejects the request, then the student can register for more courses.
- **Student View:** On the 'Registration' page itself, students can also view their past courses and their grades in it. After the instructor has finished grading their course, the course is shifted from student's active course list to student's past course list, and their grade is made visible.
- **Student/Instructor View:** Both students and instructors have an 'Archive' page. This page provides access to past courses and their associated information. This includes term, year, course name, instructor name, grade distribution, schedule etc. Basically all the information available with us in the database.
- **Instructor View:** Instructors have access to the 'Courses' page. This page has two tables: Current Courses and Past Courses. Current Courses includes a list of courses that the instructor is teaching in the current term. Past Courses includes the list of courses that the instructor has taught before. Also, during the registration window (set by admin), an instructor can also float a course, using the 'Float Course' button. The floated course will then be made available to students for registration.
- **Instructor View:** During the grading window (set by admin). The 'Grade' button, which is present in the Current Courses table, will be enabled. The instructor can click on this button and submit grades for every student registered in that course.
- **Admin View:** The Admin has access to the 'Manage' page. This page allows admin to set system configuration, like window (registration, audit, withdraw etc), year, term. It also allows the Admin to add/delete users from the CMS.

- **Admin View:** The Admin also has access to a list of all users (with searching capabilities). The Admin can use this list to search for users and provide information to them on their behalf. Also, the Admin can delete users based on their details, if necessary.

3.2 System View

We have added three types of special functionality: Materialized Views, Triggers and Indexes.

1. **Materialized View:** We created 3 Materialized Views:
   - **registrations_view**: It materializes the projection of a triple join between registrations, course_offerings and students table. It is used in fetching pending/active registrations of students in a course, and improves running time
   - **instructor_past_course_offerings**: It materializes the projection of triple join between course_offerings, sections and teachings table. It is used in fetching past courses of an instructor. It improves performance as this view is rarely refreshed (end of term). Hence, join can be avoided in general.
   - **instructor_live_course_offerings**: It materializes the projection of join between course_offerings and live_courses table. It is used in fetching live courses of an instructor. It improves performance as this view is rarely refreshed (start of term). Hence, join can be avoided in general.
2. **Triggers:** We have created 3 Triggers:
   - **registrations_trigger**: It is triggered whenever there is an INSERT/UPDATE/DELETE operation on the registrations table. It refreshes the Materialized view "registrations_view".
   - **instructor_past_course_offerings_trigger**: It is triggered whenever there is a DELETE operation on the live_courses table. It refreshes the Materialized view "instructor_past_course_offerings".
   - **instructor_live_course_offerings_trigger**: It is triggered whenever there is an INSERT/UPDATE/DELETE operation on the live_courses table. It refreshes the Materialized view "instructor_live_course_offerings".
3. **Indexes:** We created the following Indexes apart from the ones that were already present on the primary keys of the tables.
   - course_offerings_index_cuuid *on* course_offering(course_uuid)
   - courses_index *on* courses(name)
   - instructors_index_uname *on* instructors(username)
   - live_courses_index *on* live_courses(instructor_id)
   - registrations_index *on* registrations(student_id)
   - sections_index_couuid *on* sections(course_offering_uuid)
   - students_index_uname *on* students(username)
   - teachings_index *on* teachings(instructor_id)
   - terms_index *on* terms(year,term)
4. **Security:**
   - Only admin is allowed to change the important configurations including course registration window, audit/withdraw window, and grading window.
   - The admin set the year and term for the ongoing semester.
   - Only the admin has access to the list of all students and instructors.
   - Only the admin has the ability to add/delete users from the system.

Student/Instructor Queries (3.1)

```
-- queries for authentication
SELECT * FROM students WHERE username = %s AND password = %s
SELECT * FROM instructors WHERE username = %s AND password = %s
SELECT * FROM admins WHERE username = %s AND password = %s
```

```
-- query executed when a student adds a course request
INSERT INTO registrations (student_id, course_offering_uuid, instructor_id, tstatus, grade)
VALUES (%s, %s, %s, %s, %s)
```

```
-- query executed when a student removes a course request
DELETE FROM registrations
WHERE (student_id = %s AND course_offering_uuid = %s)
```

```
-- queries for updating the registrations status of a student when instructors accepts/
    rejects it
UPDATE registrations SET tstatus = 'ACCEPTED' WHERE id = %s
UPDATE registrations SET tstatus = 'REJECTED' WHERE id = %s
```

```
-- query executed when a student audits a course
UPDATE registrations SET tstatus = 'AUDIT'
WHERE (student_id = %s AND course_offering_uuid = %s)
```

```
-- query executed when a student withdraws a course
UPDATE registrations SET tstatus = 'WITHDRAW'
WHERE (student_id = %s AND course_offering_uuid = %s)
```

```
-- queries for fetching current courses of a student
SELECT id FROM students WHERE username = %s
SELECT * FROM registrations WHERE student_id = %s AND tstatus <> 'PAST'
```

```
-- queries for fetching past courses done by a student
SELECT id FROM students WHERE username = %s
SELECT * FROM registrations WHERE student_id = %s AND tstatus = 'PAST'
```

```
-- query for adding grade of a student in a course
UPDATE registrations
SET grade = %s, tstatus = 'PAST'
WHERE student_id = %s AND course_offering_uuid = %s
```

```
-- query for updating the grades of a course
INSERT INTO grade_distributions (course_offering_uuid, section_number, a_count, ab_count,
    b_count, bc_count, c_count, d_count, f_count, s_count, u_count, cr_count, n_count,
    p_count, i_count, nw_count, nr_count, other_count)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
```

```
-- queries for getting all the course_offering in a given term
SELECT * FROM terms WHERE year = %s AND term = %s
SELECT uuid, name, term_code FROM course_offerings WHERE term_code = %s
```

```
-- get course_offerings corresponding to a given course_uuid
SELECT uuid, name, term_code FROM course_offerings WHERE uuid = %s
```

```
-- query for getting the section number from course_offering_uuid
SELECT number FROM sections WHERE course_offering_uuid = %s
```

Admin Queries (3.1)

```
-- updating configuration
UPDATE config SET config = %s, year = %s, term = %s
```

```
-- adding a new student/instructor/admin
INSERT INTO students (id, name, username, password) VALUES (%s, %s, %s, %s)
INSERT INTO instructors (id, name, username, password) VALUES (%s, %s, %s, %s)
INSERT INTO admins (username, password, name) VALUES (%s, %s, %s)
```

```
-- removing a student/instructor/admin
DELETE FROM instructors WHERE username = %s
DELETE FROM students WHERE username = %s
DELETE FROM admins WHERE username = %s
```

Queries for creating Materialized Views (3.2)

```
CREATE MATERIALIZED VIEW registrations_view
AS
    SELECT registrations.id as reg_id, username, students.name as name, instructor_id,
    course_offerings.uuid as uuid, course_offerings.name as course, tstatus
    FROM registrations
    JOIN course_offerings on (course_offerings.uuid = registrations.course_offering_uuid)
    JOIN students on registrations.student_id = students.id;
```

```
CREATE MATERIALIZED VIEW instructor_past_course_offerings
AS
    SELECT course_offerings.uuid as uuid, instructor_id, term_code, course_offerings.name as
     name
    FROM course_offerings
    JOIN sections on sections.course_offering_uuid = course_offerings.uuid
    JOIN teachings on sections.uuid = teachings.section_uuid
    WHERE course_offering_uuid NOT IN (
        SELECT course_offering_uuid FROM live_courses
    );
```

```
CREATE MATERIALIZED VIEW instructor_live_course_offerings
AS
    SELECT course_offerings.uuid as uuid, instructor_id, term_code, course_offerings.name as
     name
    FROM course_offerings
    JOIN live_courses on live_courses.course_offering_uuid = course_offerings.uuid;
```

Queries for creating Triggers (3.2)

```
CREATE OR REPLACE FUNCTION refresh_registrations_view()
    RETURNS TRIGGER
    LANGUAGE PLPGSQL
AS
$$
BEGIN
    REFRESH MATERIALIZED VIEW registrations_view;
    RETURN NEW;
END;
$$;

CREATE TRIGGER registrations_trigger
    AFTER UPDATE OR INSERT OR DELETE
    ON registrations
    FOR EACH STATEMENT
EXECUTE PROCEDURE refresh_registrations_view();
```

```
CREATE OR REPLACE FUNCTION refresh_instructor_past_course_offerings()
    RETURNS TRIGGER
    LANGUAGE PLPGSQL
AS
$$
BEGIN
    REFRESH MATERIALIZED VIEW instructor_past_course_offerings;
    RETURN NEW;
END;
$$;

CREATE TRIGGER instructor_past_course_offerings_trigger
    AFTER DELETE
    ON live_courses
    FOR EACH STATEMENT
EXECUTE PROCEDURE refresh_instructor_past_course_offerings();
```

```
CREATE OR REPLACE FUNCTION refresh_instructor_live_course_offerings()
    RETURNS TRIGGER
    LANGUAGE PLPGSQL
AS
$$
BEGIN
    REFRESH MATERIALIZED VIEW instructor_live_course_offerings;
    RETURN NEW;
```

```
END;
$$;

CREATE TRIGGER instructor_live_course_offerings_trigger
    AFTER UPDATE OR INSERT OR DELETE
    ON live_courses
    FOR EACH STATEMENT
EXECUTE PROCEDURE refresh_instructor_live_course_offerings();
```

Queries for creating Indices (3.2)

```
create index if not exists students_index_uname
on students(username);

create index if not exists instructors_index_uname
on instructors(username);

create index if not exists course_offerings_index_cuuid
on course_offerings(course_uuid);

create index if not exists courses_index
on courses(name);

create index if not exists live_courses_index
on live_courses(instructor_id);

create index if not exists registrations_index
on registrations(student_id);

create index if not exists sections_index_couuid
on sections(course_offering_uuid);

create index if not exists teachings_index
on teachings(instructor_id);

create index if not exists terms_index
on terms(year, term);
```

3.3 Sample queries with run-time

```
-- query for fetching current courses taught by an instructor (17ms)
SELECT uuid, name, year, term
FROM instructor_live_course_offerings
JOIN terms on term_code = terms.code
WHERE instructor_id = %s
```

instructor_id = 3677061, time = 17ms

```
-- queries for fetching all the details of a course offering (100ms)
SELECT * FROM course_offerings WHERE uuid = %s;
SELECT * FROM terms WHERE code = %s;
SELECT * FROM sections WHERE course_offering_uuid = %s;
SELECT * FROM schedules WHERE uuid = %s;
SELECT name FROM instructors
WHERE id IN (
    SELECT instructor_id FROM teachings
    WHERE section_uuid IN (
        SELECT uuid FROM sections WHERE course_offering_uuid = %s
    )
)
SELECT * FROM grade_distributions WHERE course_offering_uuid = %s;
```

course_offering_uuid = 76b9c458-d3c2-38c4-951f-69b6900bd7fe, time = 100ms

```
-- queries executed when an instructor floats a course (89ms)
INSERT INTO courses (uuid, name, number) VALUES (%s, %s, %s);
INSERT INTO course_offerings (uuid, course_uuid, term_code, name) VALUES (%s, %s, %s, %s);
INSERT INTO schedules (uuid, start_time, end_time, mon, tues, wed, thurs, fri, sat, sun)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s);
INSERT INTO sections (uuid, course_offering_uuid, section_type, number, room_uuid,
    schedule_uuid) VALUES (%s, %s, %s, %s, %s, %s);
INSERT INTO teachings (instructor_id, section_uuid) VALUES (%s, %s);
INSERT INTO live_courses (course_offering_uuid, instructor_id) VALUES (%s, %s);
```

course_offering_uuid = f6fa6e18-92b4-47e4-aca6-55c439351d3e, Time = 89ms

```
-- query for getting the list of students who are enrolled in the current course (10ms)
SELECT reg_id, username, name, course, tstatus
FROM registrations_view
WHERE (tstatus = 'ACCEPTED' or tstatus = 'AUDIT' or tstatus = 'WITHDRAW') and instructor_id
    = %s
and uuid = %s
```

instructor_id = 3677061, course_offering_uuid = ba9134fe-4e1a-4c25-a5e7-15105ef3cabc, Time = 10ms

```
-- query for fetching past courses taught by an instructor (31ms)
SELECT uuid, name, year, term
FROM instructor_past_course_offerings
JOIN terms ON term_code = terms.code
where instructor_id = %s
```

instructor_id = 3677061, time = 31ms