طراحی پایگاه داده دانشگاهی

جهت مدیریت دانشکده‌ها، اساتید، دوره‌ها، دانشجویان و نمرات

خرداد ۱۴۰۴

فهرست مطالب

۱ مقدمه

هدف اصلی این سیستم، ارائه‌ی ساختاری منظم و کارآمد برای مدیریت و ذخیره‌سازی اطلاعات آکادمیک است.
از طرح مسئله قابل استنتاج است که سامانه‌ی دانشگاهی می‌بایست اطلاعاتی را در خصوص دانشکده‌های دانشگاه، اساتید، دانشجویان، دروس و کلاس‌ها ثبت نماید.
و همچنین روابطی همچون اخذ (رابطه‌ی دانشجو و کلاس)، تدریس (رابطه‌ی استاد و کلاس)، راهنما (رابطه‌ی استاد و دانشجو)، پیش‌نیاز (رابطه‌ی درس با درس)، تعلق (بین دانشکده و سایر موجودیت‌ها و بین دروس و کلاس‌ها) در طرح باید موجود باشد.

طرح پایگاه داده در فایل source/university_db/ddl.sql موجود است که به تفصیل در ادامه شرح داده می‌شود.

در فایل source/university_db/db_handler.py پیاده‌سازی نمونه‌ای از لایه عملیاتی دیتابیس ارائه شده است. کلاس UniversityData به عنوان یک کلاس انتزاعی (Abstract Class) طراحی گردیده است و امکان نمونه‌سازی مستقیم از آن وجود ندارد.

به جای آن، کلاس AdminQueryHandler به عنوان مثالی عملی پیاده‌سازی شد تا مجموعه‌ای از پرس‌وجوهای رایج را در قالب توابع مستقل ارائه دهد.

رویکرد بهینه‌تر، تعریف کلاس‌های جداگانه برای هر دسته از کاربران بود که به صورت ضمنی مفهوم کنترل سطح دسترسی را القا کند. با این حال، برای جلوگیری از پیچیدگی بیش از حد پروژه، از این رویکرد صرف‌نظر شد.

فایل source/university_db/__init__.py مسئول تبدیل دایرکتوری به یک پکیج پایتون است.
تنظیمات و رفتار مورد نظر برای پکیج در همین فایل تعریف شده‌اند.

فایل source/README.md حاوی راهنمای جامع استفاده از پکیج است. در این فایل روند ساخت و استفاده از یک پایگاه داده‌ی ساده شرح داده شده است. برای جلوگیری از تکرار، از بازگویی مجدد این توضیحات خودداری می‌شود.

نمودار کامل ERD نیز در همین فایل، یعنی README.md، قابل دسترس است.

۲ معرفی طراحی پایگاه داده

۲ - ۱ معرفی موجودیت‌ها

۲ - ۱ - ۱ دانشکده

موجودیت دانشکده به عنوان یکی از مهم‌ترین موجودیت‌های سیستم مدیریت دانشگاه محسوب می‌شود. دانشکده‌ها مسئولیت ارائه دروس را بر عهده دارند. وجود این موجودیت در سیستم ضروری است زیرا تمامی عناصر دیگر سیستم اعم از دانشجویان، اساتید و دروس به نوعی به دانشکده‌ها وابسته هستند. مدیریت بودجه، تعیین رئیس دانشکده، تخصیص ساختمان از طریق این موجودیت امکان‌پذیر می‌شود.

خصیصه‌ها

۱ - نام دانشکده

۲ - شماره تماس

۳ - بودجه

۴ - نام ساختمان دانشکده

۵ - نام رییس دانشکده

۲ - ۱ - ۲ دانشجو

این موجودیت نه تنها شامل اطلاعات شخصی دانشجویان است بلکه پیگیری وضعیت تحصیلی، تعداد واحدهای گذرانده شده و وضعیت فعلی آنها را نیز فراهم می‌کند. مدیریت دانشجویان بدون داشتن این موجودیت غیرممکن است. ثبت‌نام در دروس، پیگیری پیشرفت تحصیلی، محاسبه معدل، صدور مدارک و تشخیص دانشجویان واجد شرایط فارغ‌التحصیلی همگی وابسته به وجود این موجودیت هستند.

خصیصه‌ها

۱ - شماره دانشجویی

۲ - نام

۳ - نام خانوادگی

۴ - دانشکده

۵ - رشته

۶ - واحدهای گذرانده

۷ - ایمیل

۸ - تاریخ شروع تحصیل

۹ - وضعیت

۲ - ۱ - ۳ مدرس

این موجودیت شامل اطلاعات مربوط به اساتید، مربیان و کادر آموزشی دانشگاه است. مدیریت . منابع انسانی، تخصیص تدریس، پرداخت حقوق، ارزیابی عملکرد و برنامه‌ریزی آموزشی همگی نیازمند وجود این موجودیت هستند.

خصیصه‌ها

۱ - شماره هویتی (ای‌دی)

۲ - نام

۳ - نام خانوادگی

۴ - دانشکده

۵ - رتبه‌ی علمی

۶ - حقوق

۷ - ایمیل

۸ - تاریخ اشتغال به‌کار

۹ - شماره تلفن دفتر

۲ - ۱ - ۴ درس

درس محتوای آموزشی اصلی که در دانشگاه ارائه می‌شود را نمایندگی می‌کند. این موجودیت پایه و اساس برنامه‌های درسی و مناهج آموزشی است. بدون تعریف دقیق دروس، امکان ایجاد برنامه‌های تحصیلی وجود ندارد.

خصیصه‌ها

۱ - شناسه درس

۲ - عنوان درس

۳ - واحدها

۴ - دانشکده‌ی ارائه‌کننده

۵ - توضیحات

۲ - ۱ - ۵ کلاس

بخش یا کلاس نمایانگر ارائه عملی و زمان‌بندی شده یک درس در ترم خاص است. این موجودیت پل ارتباطی بین درس نظری و اجرای عملی آن محسوب می‌شود. مدیریت زمان‌بندی کلاس‌ها، تخصیص اتاق‌ها، کنترل ظرفیت کلاس‌ها و پیگیری تعداد دانشجویان ثبت‌نام شده همگی از طریق این موجودیت انجام می‌شود. بدون وجود این موجودیت، امکان برنامه‌ریزی ترم تحصیلی وجود نخواهد داشت.

خصیصه‌ها

۱ - شناسه درس

۲ - شناسه کلاس

۳ - ترم

۴ - سال

۵ - وقت و روز جلسات

۶ - ظرفیت

۷ - تعداد ظرفیت پرشده

روی این موجودیت می‌بایست یک محدودیت جامعیتی تعریف شود تا ظرفیت پرشده از کل ظرفیت بیش‌تر نشود.

۲ - ۲ معرفی روابط

۲ - ۲ - ۱ روابط تعلق

۲ - ۲ - ۱ - ۱ درس‌دانشکده

رابطه‌ی درس و دانشکده است.

یعنی: هر درس حتما متعلق به یک دانشکده بوده‌است اما یک دانشکده ممکن است یک درس ارایه داده باشد.

۲ - ۲ - ۱ - ۲ مدرس-دانشکده

۲ - ۲ - ۱ - ۳ دانشجو-دانشکده

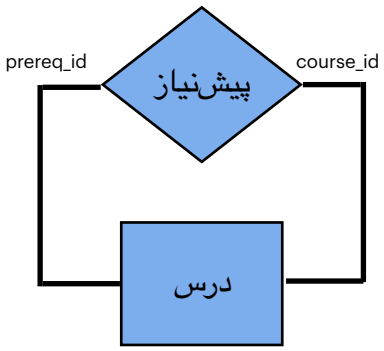
۲ - ۲ - ۱- ۴ کلاس-درس

۲ - ۲ - ۲ رابطه‌ی راهنما

بین دانشجو و مدرس برقرار است.

۲ - ۲ - ۳ رابطه‌ی پیش‌نیاز

بین درس و درس برقرار است.

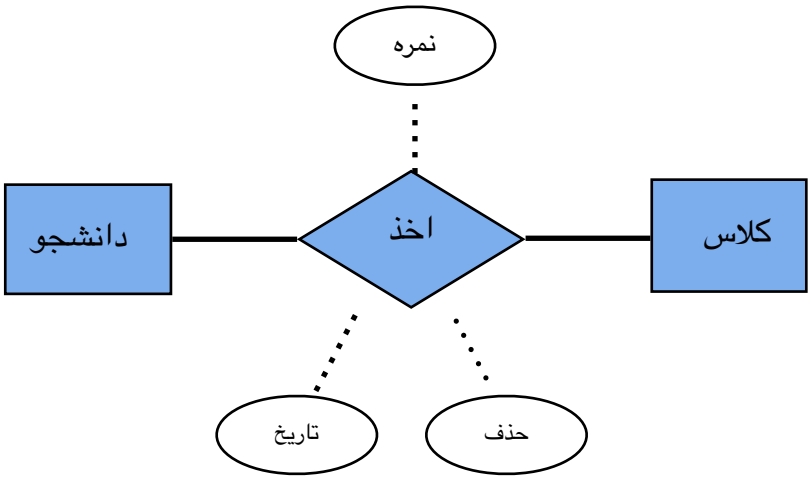


۲ - ۲ - ۴ رابطه‌ی تدریس

بین مدرس و کلاس برقرار است.

۲ - ۲ - ۵ رابطه‌ی اخذ

بین دانشجو و کلاس برقرار است و در حقیقت یک رابطه یا موجودیت نیست. بلکه یک پل است.



۲ - ۳ کاردینالیتی و درجه

رابطه	موجودیت اول	موجودیت دوم	کاردینالیتی	درجه
درس-دانشکده	درس	دانشکده	N : 1	دو
مدرس-دانشکده	مدرس	دانشکده	N : 1	دو
دانشکده-دانشجو	دانشجو	دانشکده	N : 1	دو
کلاس-درس	درس	کلاس	N : 1	دو
راهنما	دانشجو	مدرس	N : 1	دو
پیش‌نیاز	درس	درس	N : N	یک
تدریس	مدرس	کلاس	1 : N	دو
اخذ	دانشجو	کلاس	N : N	دو

۳ پیاده‌سازی ساختار پایگاه داده

موجودیت‌ها و روابط تشریح شده را به صورت زیر به جدول‌های متفاوت می‌نگاریم.

برای موجودیت دانشکده:

```
CREATE TABLE department (
    dept_name       VARCHAR(50) PRIMARY KEY,
    phone           VARCHAR(15),
    budget          NUMERIC(15, 2),
    building        VARCHAR(50),
    dean_name       VARCHAR(100)
);
```

برای موجودیت دانشجو:

CREATE TABLE student (

```
    id              INTEGER PRIMARY KEY,
    fname           VARCHAR(25) NOT NULL,
    lname           VARCHAR(25) NOT NULL,
    dept_name       VARCHAR(50) NOT NULL,
    major           VARCHAR(50),
    tot_cred        INTEGER,
    email           VARCHAR(60) UNIQUE NOT NULL,
    enrollment_date DATE DEFAULT CURRENT_DATE,
    status          VARCHAR(20) DEFAULT 'Active'
);
```

برای موجودیت مدرس:

CREATE TABLE instructor (

```
    id              INTEGER PRIMARY KEY,
    fname           VARCHAR(25) NOT NULL,
    lname           VARCHAR(25) NOT NULL,
    dept_name       VARCHAR(50) NOT NULL,
    academic_rank   VARCHAR(25),
    salary          NUMERIC(10, 2),
    email           VARCHAR(60) UNIQUE NOT NULL,
```

```
    hire_date       DATE DEFAULT CURRENT_DATE,
    office_number   VARCHAR(20)
);
```

برای موجودیت درس:

```
CREATE TABLE course (
    course_id       VARCHAR(10) PRIMARY KEY,
    title           VARCHAR(100) NOT NULL,
    credits         INTEGER NOT NULL,
    dept_name       VARCHAR(50) NOT NULL,
    description     TEXT
);
```

برای موجودیت کلاس:

```
CREATE TABLE section (
    course_id       VARCHAR(10),
    sec_id          VARCHAR(8),
    semester        VARCHAR(6),
    academic_year   INTEGER,
    time_slot       VARCHAR(20),
    room            VARCHAR(15),
    capacity        INTEGER,
    enrolled        INTEGER DEFAULT 0,
    PRIMARY KEY (course_id, sec_id, semester, academic_year)
);
```

برای روابط تعلق

رابطه‌ی ۲ - ۲ - ۱ - ۱:

```
ALTER TABLE course ADD CONSTRAINT fk_course_dept
    FOREIGN KEY (dept_name) REFERENCES department(dept_name)
    ON DELETE RESTRICT ON UPDATE CASCADE;
```

رابطه‌ی ۲ - ۲ - ۱ - ۲:

```
ALTER TABLE instructor ADD CONSTRAINT fk_instructor_dept
    FOREIGN KEY (dept_name) REFERENCES department(dept_name)
    ON DELETE RESTRICT ON UPDATE CASCADE;
```

رابطه‌ی ۲ - ۲ - ۱ -۳:

```
ALTER TABLE student ADD CONSTRAINT fk_student_dept
    FOREIGN KEY (dept_name) REFERENCES department(dept_name)
    ON DELETE RESTRICT ON UPDATE CASCADE;
```

رابطه‌ی ۲ - ۲ - ۱ - ۴:

```
ALTER TABLE section ADD CONSTRAINT fk_section_course
    FOREIGN KEY (course_id) REFERENCES course(course_id)
    ON DELETE CASCADE ON UPDATE RESTRICT;
```

محدودیت‌های جامعیتی:

```
ALTER TABLE department ADD CONSTRAINT chk_budget
    CHECK (budget >= 0);

ALTER TABLE student ADD CONSTRAINT chk_tot_cred
    CHECK (tot_cred >= 0);

ALTER TABLE student ADD CONSTRAINT chk_status
    CHECK (status IN ('Active', 'Inactive', 'Graduated',
'Suspended'));

ALTER TABLE instructor ADD CONSTRAINT chk_academic_rank
    CHECK (academic_rank IN ('Assistant Professor', 'Associate
Professor','Professor', 'Lecturer', 'Adjunct'));

ALTER TABLE instructor ADD CONSTRAINT chk_salary
    CHECK (salary >= 0);

ALTER TABLE course ADD CONSTRAINT chk_credits
    CHECK (credits > 0 AND credits <= 4);

ALTER TABLE section ADD CONSTRAINT chk_semester
    CHECK (semester IN ('Fall', 'Winter', 'Spring', 'Summer'));

ALTER TABLE section ADD CONSTRAINT chk_academic_year
    CHECK (academic_year > 1701 AND academic_year < 2100);

ALTER TABLE section ADD CONSTRAINT chk_capacity
    CHECK (capacity > 0);

ALTER TABLE section ADD CONSTRAINT chk_enrolled
    CHECK (enrolled >= 0);
```

```
ALTER TABLE section ADD CONSTRAINT chk_enrollment_capacity
    CHECK (enrolled <= capacity);
```

برای رابطه‌ی راهنما:

```
CREATE TABLE advisor (

    student_id      INTEGER,
    instructor_id   INTEGER,
    start_date      DATE DEFAULT CURRENT_DATE,
    end_date        DATE,
    PRIMARY KEY (student_id),
    FOREIGN KEY (student_id) REFERENCES student(id)
        ON DELETE CASCADE,
    FOREIGN KEY (instructor_id) REFERENCES instructor(id)
        ON DELETE SET NULL
);
```

برای رابطه‌ی پیش‌نیاز:

```
CREATE TABLE prereq (

    course_id       VARCHAR(10),
    prereq_id       VARCHAR(10),
    PRIMARY KEY (course_id, prereq_id),
    FOREIGN KEY (course_id) REFERENCES course(course_id)
        ON DELETE CASCADE,
    FOREIGN KEY (prereq_id) REFERENCES course(course_id)
        ON DELETE SET NULL
);
```

برای رابطه‌ی تدریس:

```
CREATE TABLE teaches (
    instructor_id   INTEGER,
    course_id       VARCHAR(10),
    sec_id          VARCHAR(8),
    semester        VARCHAR(6),
    academic_year   INTEGER,
    PRIMARY KEY (instructor_id, course_id, sec_id, semester,
academic_year),
    FOREIGN KEY (instructor_id) REFERENCES instructor(id)
        ON DELETE CASCADE,
```

```
    FOREIGN KEY (course_id, sec_id, semester, academic_year)
        REFERENCES section ON DELETE CASCADE
);
```

محدودیت‌ها:

```
ALTER TABLE teaches ADD CONSTRAINT chk_teaches_semester
    CHECK (semester IN ('Fall', 'Winter', 'Spring', 'Summer'));

ALTER TABLE teaches ADD CONSTRAINT chk_teaches_year
    CHECK (academic_year > 1701 AND academic_year < 2100);
```

برای رابطه‌ی اخذ:

```
CREATE TABLE takes (
    student_id      INTEGER,
    course_id       VARCHAR(10),
    sec_id          VARCHAR(8),
    semester        VARCHAR(6),
    academic_year   INTEGER,
    cancelled       BOOLEAN DEFAULT FALSE,
    grade           VARCHAR(2),
    enrollment_date DATE DEFAULT CURRENT_DATE,
    PRIMARY KEY (student_id, course_id, sec_id, semester,
academic_year),
    FOREIGN KEY (student_id) REFERENCES student(id)
        ON DELETE CASCADE,
    FOREIGN KEY (course_id, sec_id, semester, academic_year)
        REFERENCES section ON DELETE CASCADE
);
```

محدودیت‌ها:

```
ALTER TABLE takes ADD CONSTRAINT chk_takes_semester
    CHECK (semester IN ('Fall', 'Winter', 'Spring', 'Summer'));

ALTER TABLE takes ADD CONSTRAINT chk_takes_year
    CHECK (academic_year > 1701 AND academic_year < 2100);

ALTER TABLE takes ADD CONSTRAINT chk_grade
    CHECK (grade IN ('A+', 'A', 'A-', 'B+', 'B', 'B-',
           'C+', 'C', 'C-', 'D+', 'D', 'F'));
```

البته این تعاریف به‌صورت سرهم در فایل سورس موجود است.