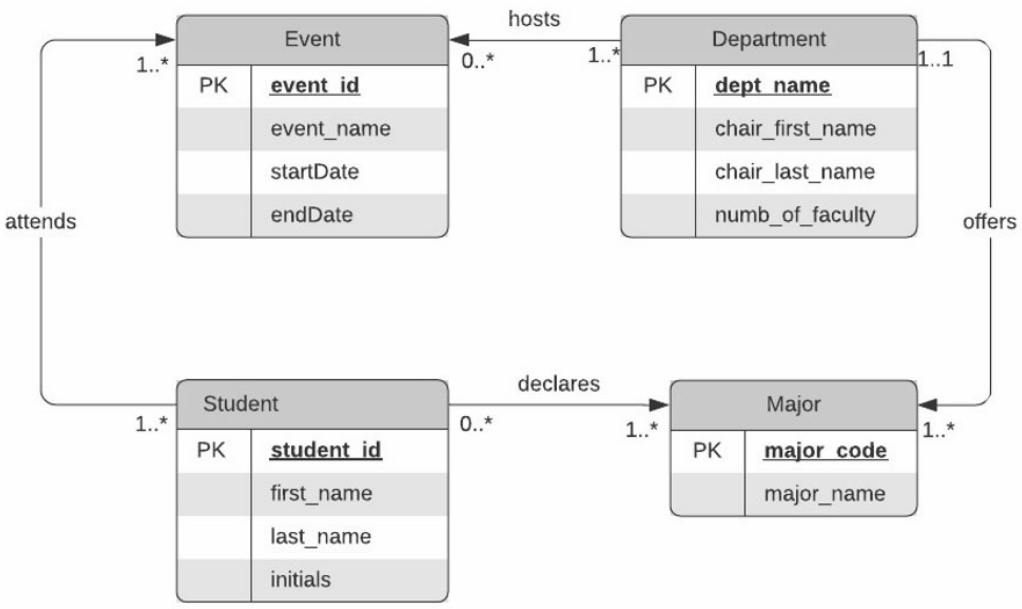
CSC423- Database Systems
Taumel Vega

Project Part 2

Conceptual ER diagram from Part 1



2. Develop a logical data model based on the following requirements: (11/19/21)

a) Derive relations from conceptual model

- Department
- Major
- Student
- Event
- 1 : * Department→ Major (needs FK on the many side)
    - Major (**major_code**, major_name, dept_id)
- * : * Student→ Major (many-to-many, needs new table)
    - Declaring_major (stu_id,major_code)
- * : * Student→ Event (many-to-many, needs new table)
    - Attending_event (stu_id, event_id)
- * : * Department→ Event (many-to-many, needs new table)
    - Hosting_event (dept_id, event_id)

1. Department (**dept_id**, department_name, chair_first_name, chair_last_name, numb_of_faculty)
    - Primary Key: dept_id
    - Alternate Key: department_name
2. Major (**major_code**, major_name, dept_id)
    - Primary key: dept_id
    - Alternate Key: major_name
    - Foreign Key: dept_id references Department(dept_id)

3. Student (**stu_id**, first_name, last_name, initials)
    - Primary Key: stu_id
4. Event (**event_id**, event_name, startDate, endDate)
    - Primary key: event_id
    - Alternate Key: event_name
5. Hosting_event(**dept_id**, **event_id**)
    - Primary Key: dept_id, event_id
    - Foreign Key: dept_id references Department(dept_id)
    - Foreign Key: event_id references Event(event_id)
6. Attending_event(**stu_id**, **event_id**)
    - Primary Key: stu_id, event_id
    - Foreign Key: stu_id references Student(stu_id)
    - Foreign Key: event_id references Event(event_id)
7. Declaring_major(**stu_id**,**major_code**)
    - Primary key: stu_id, event_id
    - Foreign key: stu_id references Student(stu_id)
    - Foreign key: major_code references Major(major_code)

b) Validate Normalization

- 1NF -- None of the tables (1-7 above) contain repeating groups; and the intersection of each row and column contains only one value.
- 2NF --There are no partial dependencies.
- 3NF – On the *Student* relation there is a transitive dependency *(first_name, last_name → initials)*. However, creating an additional table introduces complexity to the database because a join would be required to find the person's initials. In addition, the redundancy removed by the additional table would be minimal in this particular case. Therefore, I decided to keep the *initials* attribute in the *Student* table

c) Validate the logical model against user transactions

1. List the details of students that are attending a named event.
    - The details of students are held in *Student* entity and the details of events are held in *Event* relationship. We can use the *Attending_event* relationship to list students attending a specific event.
2. Count the number of Majors offered by department. List major count and department name
    - We can get the count from the *Major* entity, which lists the Majors offered by department. Using dept_id we can get the department name from the *Department* entity.
3. List events being hosted by named department.
    - The event's details are held in the *Event* entity. And the department's details are held in the *Department* entity. From the *hosting_event* entity we can get the events hosted by a named department.
4. List the chair name for every department.
    - The *department* entity contains the chair name.
5. Find the events being attended by a named student and also lists the same student's major (or majors).

- The *Student* entity contains student's name. The *Event* entity contains all Events details and the *Major* entity contains the Major name.
- From the *Attending_event* and *Declaring_major* entities we can get the events and majors corresponding to a named student.

d) Integrity contraints

i. PK constraints

- Major(major_code),
- Department(dept_id)
- Student(stu_id)
- Event(event_id)

must be unique and not null. Also include domain constraints. (see ii Below)

ii. FK constraints

- Major (dept_id)
- Hosting_event (dept_id)
- Hosting_event (event_id)
- Attending_event (stu_id)
- Attending_event (event_id)
- Declaring_major (stu_id)
- Declaring_major (event_id)

Must be unique and not null because in addition to FKs they are also PKs. These tables were derived from many-to-many relationships

iii. AK constraints

- Major(major_name) unique and not null
- Department(department_name) unique and not null

iv. General constraints

- Event(startDate) less than Event(endDate)
- Event(startDate) greater than current date
- Department(Department_name) must start with *Department*

ii. Domain constraints

- Major(major_code) must be 3 characters
- Student(initials) greater than 1 char
- Student(stu_id) must be 9 digits
- Event(event_id) must be 5 chars

e) E-R diagram, logical level