University DBMS Conceptual Design

2. Develop a logical data model based on the following requirements:

ASSUMPTIONS:

- **Students can return to the same school to complete multiple majors over their lifetime, no reason to give a cap to the number of majors a student can declare.**
- **No two departments will have the same name**
- **No two majors will have the same name**
- **Students need a school-provided non-duplicative ID to be told apart in the system since students with the same major and name/initials can easily exist**
- **No two events with the same name can start on the same day**
- **The same professor cannot be a chair of more than 1 department**

a. Derive relations from the conceptual model.

Based on the multiplicities in this table:

Entity 1	Relationship	Entity 2	Participation	Cardinality	Multiplicity	Type of Relationship
Student	Declare	Major	1	*	1..*	
Major	Define	Student	1	*	1..*	* : *
Major	Reference	Department	1	1	1..1	
Department	Clafisy	Major	1	*	1..*	1 : *
Department	Host	Event	0	*	0..*	
Event	Utilize	Department	1	*	1..*	* : *
Student	Attend	Event	1	*	1..*	
Event	Entertain	Student	1	*	1..*	* : *

We can see there is only a singular 1 : * relationship. Therefore

- **Major** has a foreign key from **Department**

Also have to make three new tables from our * : * relationships:

- **DeclaredMajor** that relates Student to Major
- **EventAttendance** that relates Student to Event
- **UtilizedDepartment** that relates an Event to a Department

b. Validate the logical model using normalization to **3NF**

Department(depName, chairName, numFaculty)

- Since multiple chairs can have the same name, numerous departments can have the same number of faculty working within them (i.e. neither can be primary keys), and the chair name does not relate to a number of faculty at all, **Department** is in 3rd normal form.

Student(studentID, studName, studInitials)

- Since multiple students can have the same name, and even more students can have the same initials (i.e. neither can be primary keys), and prevent constant, repetitive joining to fix the transitive dependency between studName and studIntials, **Student "**is in" 3rd normal form.

Event(eventName, startDate, endDate)

- Since endDate is an Alternate Key, and that covers all attributes in the table, **Event** is in 3rd normal form.

Major(majorName, *Code*) *Code* from **DepCode**
DepCode(Code, *depName*) *depName* from **Department**

- While majorName determines both Code and depName, Code also determines depName, which is a transitive dependency. The table needed to change - from all three attributes in one **Major** table - to the above **Major** and **DepCode** tables. **Major** and **DepCode** now only have a PK and one attribute so no other relations exist, so both are in 3rd normal form.

DeclaredMajor(*studentID, majorName*) *studentID* from **Student**, *majorName* from **Major**

- All attributes are PK's, so no other relations exist. **DeclaredMajor** is in 3rd normal form

EventAttendance(*studentID, eventName, startDate*) *studentID* from **Student**, *eventName* & *startDate* from **Event**

- All attributes are PK's, so no other relations exist. **EventAttendance** is in 3rd normal form

UtilizedDepartment(*eventName, startDate, depName*) *eventName* & *startDate* from **Event**, *depName* from **Department**

- All attributes are PK's, so no other relations exist. **UtilizedDepartment** is in 3rd normal form

c. Validate the logical model against user transactions

- If the user wants to see all students that attened an event, they can search for all unique studentIDs in the **EventAttendance** table and use those IDs to find names in the **Student** table.

- If the user wants to count how many students are in each major they can figure that out just counting how many times a major shows up in the **DeclaredMajor** table.

- If the user wants to find each event the chemistry department did only knowing the code for the department, they can get the depName using the code in the **DepCode** table and use find all events in the **UtilizedDepartment** table that matches that depName.

- If the user wants to find the number of majors a specific department offers, they can get the code of that department using the **DepCode** table and find all majors that have a matching Code.

- If the user wants to find all events that have not started yet they can just look in the **Event** table and find all events where the startDate is greater than the current date.

d. Define integrity constraints for {Primary, *Foreign*, Alternate, General} keys

Department(depName, chairName, numFaculty)

- depName cannot be NULL since it is a primary key, and must begin with the characters “Department”. No other attributes have constraints.

Student(studentID, studName, studInitials)

- studentID cannot be NULL since it is a primary key. studInitials must be more than one characer long. No other attributes have constraints.

Event(eventName, startDate, endDate)

- eventName and startDate cannot be NULL since they are primary keys. startDate must be later than the current date. endDate must be lower than the startDate.

Major(majorName, *Code*) *Code* from **DepCode**

- majorName cannot be NULL since it is a primary key. *Code* should come from the **DepCode** table.

- If a *Code* is ever deleted the assciated major should be deteted, as that implies a department was deleted, as such the Univesrity would probably no longer offer all associated majors. If it gets updated, the university probably moved the major’s code to a new department and that change should be reflected.

DepCode(Code, *depName*) *depName* from **Department**

- Code cannot be null since it is a primary key. All Codes also need to be exactly three characters long. *depName* should come from the **Department** table.

- If *depName* is deleted, the whole code should be deleted, as there is no department for the code to label now. If it gets updayed, the code is simply associated with a new department, and that change should be reflected.

DeclaredMajor(*studentID, majorName*) *studentID* from **Student**, *majorName* from **Major**

- studentID and majorName cannot be NULL since they are primary keys. *studentID* should come from the **Student** table. *majorName* should come from the **Major** table.

- If *studentID* is deleted, the associated declaredMajor tuple should be deleted as this implies the student is no longer associated with the university. If its updated, for some reason a student needed an updated ID, and that change should be reflected.

- If a majorName is deleted, set it to Undeclared in this table as the student simply needs to pick a new major. If it gets updated, simply update the table as the student simply requested a new major and that change should be reflected.

EventAttendance(*studentID, eventName, startDate*) *studentID* from **Student**, *eventName* & *startDate* from **Event**

- studentID, eventName, and startDate cannot be NULL since they are primary keys. *studentID* should come from the **Student** table. *eventName* and *startDate* should come from the **Event** table.

- If *studentID* is deleted, it should be set to a default 0 as, for data collection, even if a student is no longer associated with a univeristy it can be good to have data on raw number of students attending events. If it gets updated, simply update it here too as the student simply got an updated ID and that change should be reflected.

- If *eventName* or *startDate* is deleted, the associated attributes in EventAttendance tuple can be or set to a default “eName” or 1/1/2000 values. Since students need to attend one event, their studentID still needs to be associated with an event they went to, even if for some

reason that event is deleted. If either get updated, it is safe to update those values.

UtilizedDepartment(*eventName, startDate, depName*) *eventName* & *startDate* from **Event**, *depName* from **Department**

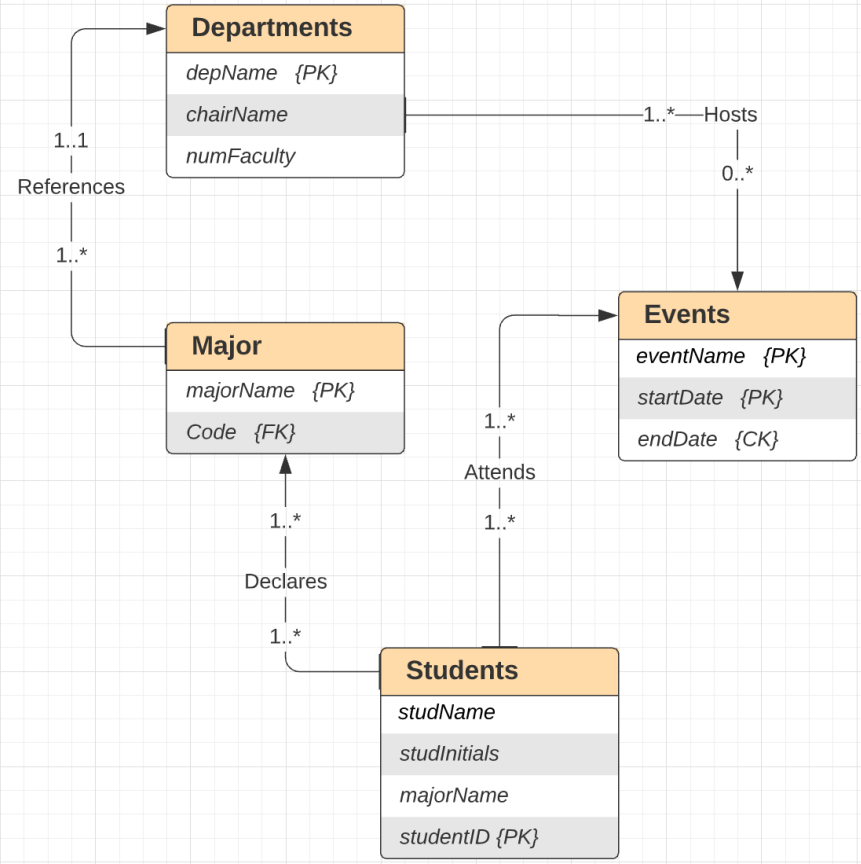
- depName, eventName, and startDate cannot be NULL since they are primary keys. *depName* should come from the **Department** table. *eventName* and *startDate* should come from the **Event** table.

- If *depName* is deleted, the associated UtilizedDepartment tuple should either be deleted as it does not benefit to keep what events they helped with. If it gets updated, the department just changed names and that change should be reflected.

- If *eventName* or *startDate* is deleted, the associated attributes in can either be simply deleted, as the event can be assumed to no longer be happening and there is no reaon to keep track of the raw number of events a department helps with. If either get updated, those changes should be updated and reflected.

University DMBS Logical Data Design

Jeffrey Hudak | November 18, 2021



Assumptions:

Students can return to the same school to complete multiple majors over their lifetime, no reason to give a cap to the number of majors a student can declare.

No two departments will have the same name

No two majors will have the same name

Students need a school-provided non-duplicative ID to be told apart in the system since students with the same major and name can easily exist

No two events with the same name can start on the same day

The same professor cannot be a chair of more than 1 department