Gabriela Serrano Echenagucia

CSC 423 – Project 2: Logical Design

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

a. Derive relations from the conceptual model.

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

c. Validate the logical model against user transactions.

d. Define integrity constraints:

i. Primary key constraints.

ii. Referential integrity/Foreign key constraints.

iii. Alternate key constraints (if any).

iv. General constraints (if any).

e. Generate the E-R diagram for the logical level (contains FKs as attributes)

A. Derive relations from the conceptual model

Department (deptNo, deptName, chairName, faculty_count)

Primary Key deptNo

Candidate Key deptName

Student (studentID, studentName, student_initials)

Primary Key studentID

Major (majorCode, majorName)

Primary Key majorCode

Candidate Key majorName

Event (eventNo, eventName, start_date, end_date)

Primary Key eventNo

Entity 1	Relationship	Entity 2	Participation	Cardinality	Multiplicity	Type of Rel.
Department	**Hosts**	Event	0	*	*..*	Many-to-many
Event	IsHostedBy	Department	1	*		
Student	**Attends**	Event	1	*	*..*	Many-to-many
Event	Comprises	Student	1	*		
Student	**Declares**	Major	1	*	*..*	Many-to-many
Major	DeclaredBy	Student	1	*		
Major	**References**	Department	1	1	1..*	One-to-many
Department	Offers	Major	1	*		

ONE-TO-MANY RELATIONSHIPS (1:*)

Major **References** Department

It becomes:

Major (majorCode, majorName, deptNo)

Primary Key majorCode

Candidate Key majorName

Foreign Key deptNo references Department(deptNo)

MANY-TO-MANY RELATIONSHIPS (*:*)

Department **Hosts** Event

Create new entity:

DeptEvent (deptNo, eventNo)

Primary Key deptNo, eventNo

Foreign Key deptNo references Department(deptNo)

Foreign Key eventNo references Event(eventNo)

Student **Attends** Event

Create new entity:

StudentAttendance (studentID, eventNo)

Primary Key studentID, eventNo

Foreign Key studentID references Student(studentID)

Foreign Key eventNo references Event(eventNo)

Student **Declares** Major

Create new entity:

StudentMajor (studentID, majorCode)

Primary Key studentID, majorCode

Foreign Key studentID references Student(studentID)

Foreign Key majorCode references Major(majorCode)

Overall,

Department (*deptNo*, deptName, chairName, faculty_count)

Student (*studentID*, studentName, student_initials)

Major (*majorCode*, majorName, deptNo)

Event (*eventNo*, eventName, start_date, end_date)

DeptEvent (*deptNo, eventNo*)

StudentAttendance (*studentID, eventNo*)

StudentMajor (*studentID, majorCode*)

B. Validate the logical model using normalization to 3NF

The proposed logical model is already in 3NF.

UNF to 1NF: Done. Table already flattened. There are no repeating groups

1NF to 2NF: Done. Partial dependencies were removed.

Primary Key dependencies:

studentID → studentName, student_initials

majorCode → majorName, deptNo

eventNo → eventName, start_date, end_date

2NF to 3NF. Check:

Department (*deptNo*, deptName, chairName, faculty_count) is in 3NF?

Yes, there are no transitive dependencies on the relation

Student (*studentID*, studentName, student_initials) is in 3NF?

No, there exists a transitive dependency studentName → student_initials. However, the additional complexity that generating a new relation from this dependency brings it's not worth it. Instead, we can just keep student_initials in the Student relation (values are three characters) and avoid extra calculations that joining the two tables would imply

---

Major (*majorCode*, majorName, deptNo) is in 3NF?

Yes, there are no transitive dependencies on the relation

Event (*eventNo*, eventName, start_date, end_date) is in 3NF?

Yes, there are no transitive dependencies on the relation

DeptEvent (*deptNo, eventNo*) is in 3NF?

Yes, the relation was created due to a many-to-many relationship and has no transitive dependencies

---

StudentAttendance (*studentID, eventNo*) is in 3NF?

Yes, the relation was created due to a many-to-many relationship and has no transitive dependencies

C. Validate the logical model against user transactions.

Describing the transactions method:

(Transaction 1): List all details of students that declared 'BIO' (or any other) as a major

In this case, we can use the StudentMajor relation to produce the required list. First, we would find the matches for the major code 'BIO' and then we can access the details for all students with a join with the Student entity which holds all information about students

---

(Transaction 2): List all details of the majors that belong to a specific department

The details of majors are held in the Major entity and the details of the department (who provides that major) are held in the Department entity. In this case, we can use the Major References Department relationship to produce the required list

---

(Transaction 3): List all the events that a student may attend to and display all details for that event

We can use the StudentAttendance relation to produce the required list. First, we would find the matches for the studentID and then we could gain access to the details for each event by doing a join with Event relation

---

(Transaction 4): List all the events that are hosted by a specific department

Using DeptEvent relation we can find the matches for the desired department, and it would list all the events hosted by that department. We could access either more details of the department by referencing the Department entity, more details about the event by referencing the Event entity, or both

---

(Transaction 5): List the departments where a student belongs. The department where a student belongs depends on their major

Using StudentMajor relation we can access information of a major declared by a certain student. To know what department of the university a student belong we would match the major to the department where it belongs. We would then join with the Major relation and use the foreign key deptNo. If we want more detail about the department we can do a join with the Department relation

D. Define integrity constraints.

i. Primary key constraints (Entity integrity): the following attributes cannot hold null values because we wouldn't be able to uniquely identify a tuple in each relation

- *deptNo* in the Department relationship
- *studentID* in the Student relationship
- *majorCode* in the Major relationship
- *eventNo* in the Event relationship
- *deptNo* and *eventNo* in the DeptEvent relationship
- *studentID* and *eventNo* in the StudentAttendance relationship
- *studentID* and *majorCode* in the StudentMajor relationship

ii. Referential integrity/Foreign key constraints: if there is a relation that contains a foreign key, all values from that attribute on the original relationship must be linked or set to null:

- deptNo in Major entity must match the values of deptNo on the Department entity, if not, all values for that attribute set to null
- deptNo and eventNo in DeptEvent relation must match the values of deptNo on the Department entity and the values of eventNo on the Event entity. They cannot be set to null because we cannot uniquely identify tuples without these values (they form a composite primary key)
- studentID and eventNo in StudentAttendance relation must match the values of studentID on the Student entity and the values of eventNo on the Event entity. They cannot be set to null because we cannot uniquely identify tuples without these values (they form a composite primary key)
- studentID and majorCode in StudentMajor relation must match the values of studentID on the Student entity and the values of majorCode on the Major entity. They cannot be set to null

because we cannot uniquely identify tuples without these values (they form a composite primary key)

iii. Alternate key constraints (if any).

- The candidate key deptName becomes alternate key when selecting deptNo as a primary key. deptName would've worked as a primary key because it is assumed that every department is named differently

iv. General constraints (if any).

- In the *Event* relation the values for the attribute *start_date* should be before the values attribute *end_date*. Also, information pertaining to events are stored ahead of time, therefore at the time of insertion an event cannot be a past date or the current date
- The major code (majorCode) must be 3 characters
- Department names (deptName) must start with "Department"
- Initials (student_initials) must be more than one character long

E. Generate the E-R diagram for the logical level (contains FKs as attributes)