**Step 1**

1. Develop a conceptual data model reflecting the following requirements: (11/02/21)

**Step 1.a**

Identify the main entity types:

|  |  |  |  |
| --- | --- | --- | --- |
| Entity Name | Description | Aliases | Occurence |
| Department | General term describing all department encompassed by Redwood University | Department | Each department has a chair. A department offers one or more major. Each department can host events |
| Student | General term describing the names of students | Student | Each student has declared at least one major. Each student must attend one or more events |
| Major | General term describing all majors | Major | Each major references one department. Major can be declared by one or more students |
| Event | General term describing all events | Event | Each event can be hosted by one or more departments. Each event will comprise one or more students |

**Step 1.b**

Identify the main relationship types between the entity types identified in "a":

Department offers Major

Department hosts Event

Major references Department

Student declares Major

Student attends Event

Event comprises Student

Event hosted Department

**Step 1.c**

Determine the multiplicity constraints for each relationship identified in "b":

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity Name | Multiplicity | Relationship | Multiplicity | Entity Name |
| Department | 1..1  1..\* | Offers  hosts | 1..\*  0..\* | Major  Event |
| Student | 1..\*  1..\* | Declares  attends | 1..\*  1..\* | Major  Event |

**Step 1.d**

Identify attributes and associate them with entity or relationship types:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entity Name | Attributes | Description | Data Type & Length | Nulls | Multi-valued |
| Department | dNo  dName  facultyCount  cName  cFirst  cLast | Uniquely identifies member of Department  Name of Department  Number of faculty  First name of chair  Last name of chair | 2 number character  30 variable characters (starts with Department)  3 number characters  15 variable characters  15 variable characters | No  No  No  No  No | No  No  No  No  No |
| Major | mName  mCode | Name of the major  Code of the major | 30 variable characters  3 variable characters  (must be 3 characters) | No  No | No  No |
| Student | sNo  sName  sFirst  sLast  sInitial | Uniquely identifies a student  First name of student  Last name of student  Initials of student | 9 variable characters  15 variable characters  15 variable characters  5 variable characters  (more than one character long) | No  No  No  No | No  No  No  No |
| Event | eNo  eName  startDate  endDate | Uniquely identifies an event  Name of an event  Start date of an event  End date of an event | 9 variable characters  100 variable characters  Date  (more than the current date)  Date  (more than the start date) | No  No  No  No | No  No  No  No |

**Step 1.e**

Determine candidate and primary key attributes for each (strong) entity type:

For Department entity:

The candidate keys are Department number (dNo), Department name (dName) Assuming department names are unique.

The primary key would be dNo since it is the shortest and the easiest to read. The other two are alternate keys.

Faculty count cannot be a candidate key since departments can have the same number of faculty.

Chair name cannot be candidate key since chair’s could have same names

For Major entity:

The candidate keys are Major name (mName) and Major code (mCode). Assuming major names and codes are unique.

The primary key would be mCode since it’s the shortest and the easiest to read. The other is an alternate key.

For Student entity:

The candidate and primary key are student number (sNo). The other attributes cannot uniquely identify a student. Students can have same names and initials.

For event entity:

The candidate and primary key are event number (eNo). The other attributes cannot uniquely identify an event. An event can have the same name if it a repeated event. Different events can occur at the same time. So, start date and end date cannot be candidate keys.

**Step 1.f**

Generate the E-R diagram for the conceptual level (no FKs as attributes):

**Diagram

Description automatically generated**

**Step 2**

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

**Step 2.a**

Derive relations from the conceptual model.

1. *Department*

Department (dNo, dName, facultyCount, cName)

Primary Key dNo

Alternate Key dName

Assuming Department names are unique

1. *Major uses dNo to reference a major to a department*

Major (mCode, mName, dNo)

Primary Key mCode

Alternate Key mName

Foreign Key dNo references Department(dNo)

Assuming Major names are unique

1. *Student*

Student (sNo, sName, sInitial)

Primary Key sNo

1. *New relation that shows student list in each major or it can show the major(s) each student declared*

StudentList (mCode, sNo)

Primary Key mCode, sNo

Foreign Key mCode references Major(mCode)

Foreign Key sNo references Stuednt(sNo)

1. *Event uses*

Event (eNo, eName, stardDate, endDate)

Primary Key eNo

1. *New relation that shows the attendance of an event*

Attendance (eNo, sNo)

Primary Key eNo, sNo

Foreign Key eNo references Event(eNo)

Foreign Key sNo references Student(sNo)

1. *New relation that shows the event(s) hosted by department(s)*

Hosting (eNo, dNo)

Primary Key eNo, dNo

Foreign Key eNo references Event(eNo)

Foreign Key dNo references Department(dNo)

**Step 2.b**

Validate the logical model using normalization to 3NF.

To get 3NF our tables must pass 1NF and 2NF

Denormalizing Data:

All the cells in each respective tables have no null value

1NF:

Each respective table has primary key(s)

*All attributes must be atomic and no multi valued or composite attributes:*

sName in Student and cName in Department have non atomic values where the names separate into first name and last name. So, we are getting rid of {cFirst, cLast} in Department table and only leaving cName. Similarly, we are getting rid of {sFirst, sLast} in Student table and only leaving sName.

The result:

Student (sNo, sName, sInitial)

Primary Key sNo

Department (dNo, dName, facultyCount, cName)

Primary Key dNo

Now, our tables pass 1NF

2NF:

All non-key attributes must be dependent on the full primary key:

Here we assume sInitial in Student has transitive dependency on sName. We could create a new table to store sName and sInitial where the sName are unique, but this adds up a lot of complexity compared to the reduncancy it removes. So, we are keeping student table as it is.

And endDate is not dependent on startdDate (User’s specification was a constraint rather than a dependency)

So, our tables pass 2NF

3NF:

Must satisfy 2NF

No attributes must depend on other attributes, that are not primary keys

Our tables satisfy all the definitions for 3NF. Thus, we can validate the logical model using 3NF normalization.

**Step 2.c**

Validate the logical model against user transactions:

Transaction (1): List the majors and the departments they belong to.

The details of the majors and its department number are stored in Major entity. Where the department number is a foreign key which references to Department entity.

Transaction (2): List the majors the students have declared:

Through the relation of Student and Major entities, we have created a relation StudentList. Here the (sNo, mCode) acts as both primary key and foreign key. This relation can show the students and their major(s) they have declared. We can list the major(s) in Student table, but we would have redundant data Student (sName, sInitial) where these would be repeated unnecessarily if we don’t need them.

Transaction (3): List the events that students have attended

Similar to transaction 2, we have created a new relation Attendance where sNo from Student entity and eNo from Event entity act as both primary and foreign keys. Without the repeating data from both tables, we can list the events that student has attended.

Transaction (4): List the events that departments have hosted

Similar to transaction 2 and 3, we have created a new relation Hosting where dNo from Department entity and eNo from Event entity act as both primary and foreign keys. Without the repeating data from both tables, we can list the event(s) hosted by department(s)

**Step 2.d**

Define integrity constraints:

1. Primary key constraints.

Primary keys cannot be null:

Department (dNo, dName, facultyCount, cName)

Primary Key dNo NOT NULL

Alternate Key dName

Major (mCode, mName, dNo)

Primary Key mCode NOT NULL

Alternate Key mName

Foreign Key dNo references Department(dNo)

Student (sNo, sName, sInitial)

Primary Key sNo NOT NULL

StudentList (mCode, sNo)

Primary Key mCode, sNo NOT NULL

Foreign Key mCode references Major(mCode)

Foreign Key sNo references Stuednt(sNo)

Event (eNo, eName, stardDate, endDate)

Primary Key eNo NOT NULL

Attendance (eNo, sNo)

Primary Key eNo, sNo NOT NULL

Foreign Key eNo references Event(eNo)

Foreign Key sNo references Student(sNo)

1. Referential integrity/Foreign key constraints.
2. Since a major must always have a department it will cascade on both update and delete

Major (mCode, mName, dNo)

Primary Key mCode NOT NULL

Alternate Key mName

Foreign Key dNo references Department(dNo) ON UPDATE CASCADE ON DELETE CASCADE

1. If student is deleted (graduated or transferred) we don’t need their major declaration anymore. And if a major is deleted, you cannot declare it anymore. So, cascade on both delete and update.

StudentList (mCode, sNo)

Primary Key mCode, sNo NOT NULL

Foreign Key mCode references Major(mCode) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key sNo references Stuednt(sNo) ON UPDATE CASCADE ON DELETE CASCADE

1. If a student is deleted, we do no action since they have already attended. And if we need the attendance number for future usage. So, we assume no action. If an event is deleted, we cascade since we delete it knowing the data of that event is not necessary.

Attendance (eNo, sNo)

Primary Key eNo, sNo NOT NULL

Foreign Key eNo references Event(eNo) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key sNo references Student(sNo) ON UPDATE CASCADE ON DELETE NO ACTION

1. Alternate key constraints (if any).
2. dName always starts with “Department” and cannot be null

Department (dNo, dName, facultyCount, cName)

Primary Key dNo NOT NULL

Alternate Key dName STARTS WITH “Department” NOT NULL

1. mName cannot be null

Major (mCode, mName, dNo)

Primary Key mCode NOT NULL

Alternate Key mName NOT NULL

Foreign Key dNo references Department(dNo)

1. General constraints (if any).
2. Student(sInitial) must be more than one character long
3. Major(mCode) must be 3 characters long
4. Event (startDate, endDate) endDate cannot be a date before startDate

startDate cannot be past or current date

**Step 2.e**

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

Diagram

Description automatically generated

**Step 3**

Translate the logical data model for the Oracle Enterprise DBMS

**Step 3.a**

Develop SQL code to create the entire database schema, reflecting the constraints identified in previous steps.

Department table:

Text

Description automatically generated

Major table:

Text

Description automatically generated

Student Table:

Text

Description automatically generated

Event Table:

**Text

Description automatically generated**

**Many to many relation creation:**

StudentList table:

Text

Description automatically generated

Attendance table:

Text

Description automatically generated

Hosting table:

Text

Description automatically generated

**Step 3.b**

Create at least 5 tuples for each relation in your database.

**Department:**

**Code:**

**Text

Description automatically generated**

**Result:**

**Text

Description automatically generated**

**Major:**

**Code:**

**Text

Description automatically generated**

**Result:**

**Text

Description automatically generated**

**Student:**

**Code:**

**Text

Description automatically generated**

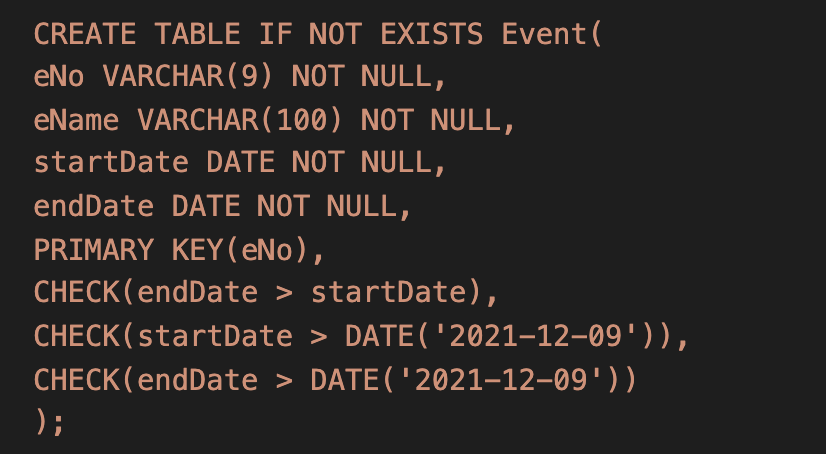
**Result:**

**A black screen with white text

Description automatically generated with medium confidence**

**Event:**

**Code:**

****

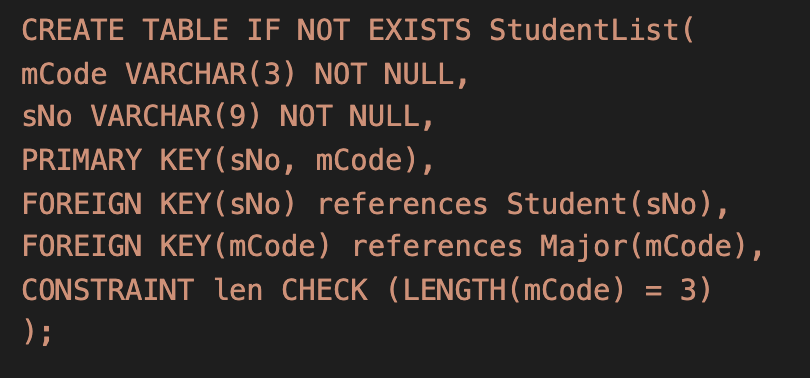
**Result:**

**Text

Description automatically generated**

**StudentList:**

**Code:**

****

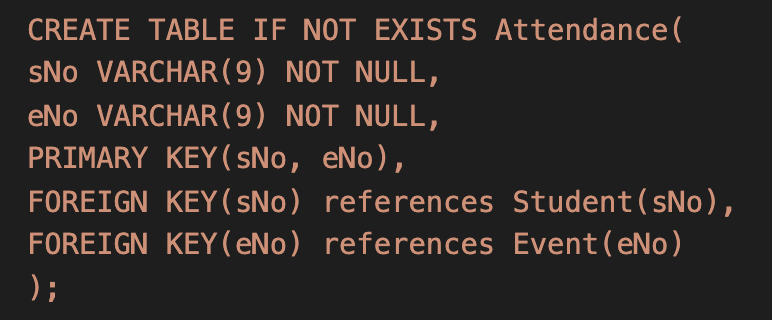
**Result:**

**Text

Description automatically generated**

**Attendance:**

**Code:**

****

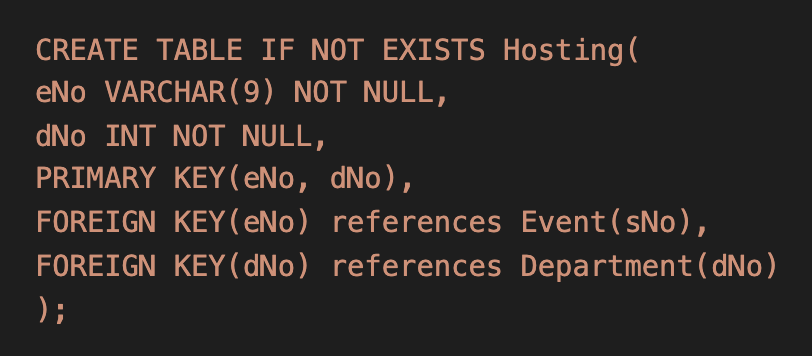
**Result:**

**Text

Description automatically generated**

**Hosting:**

**Code:**

****

**Result:**

**Text, chat or text message

Description automatically generated**

**Step 3.c**

Develop 5 SQL queries using embedded SQL (see Python tutorial).

1. Get student id, name and major

Code:

Text

Description automatically generated

Result:

Text

Description automatically generated

1. Show majors and the departments they belong to

Code:

Text

Description automatically generated

Result:

Text

Description automatically generated

1. List student id, name and events they attended

Code:

Text

Description automatically generated

Result:

Text

Description automatically generated

1. List events and the departments it is being hosted by

Code:

Text

Description automatically generated

Result:

Graphical user interface, text, application

Description automatically generated

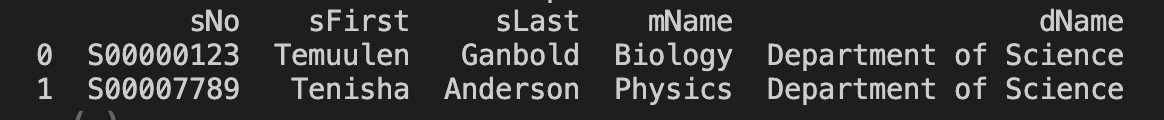
1. List the students in “Department of Science” and also show their majors

Code:

Text

Description automatically generated

Result:



**Step 3.d**

Githublink:

<https://github.com/tem-gan/csc423.git>