DATA SCI 7030: Database and Analytics

Tozammel Hossain
Oct 29, 2020

Data Science & Analytics
University of Missouri

Agenda

- Workflow in a Database Design
  - Requirement analysis (done)
  - The Entity-Relationship Model (done)
  - ERD to DDL

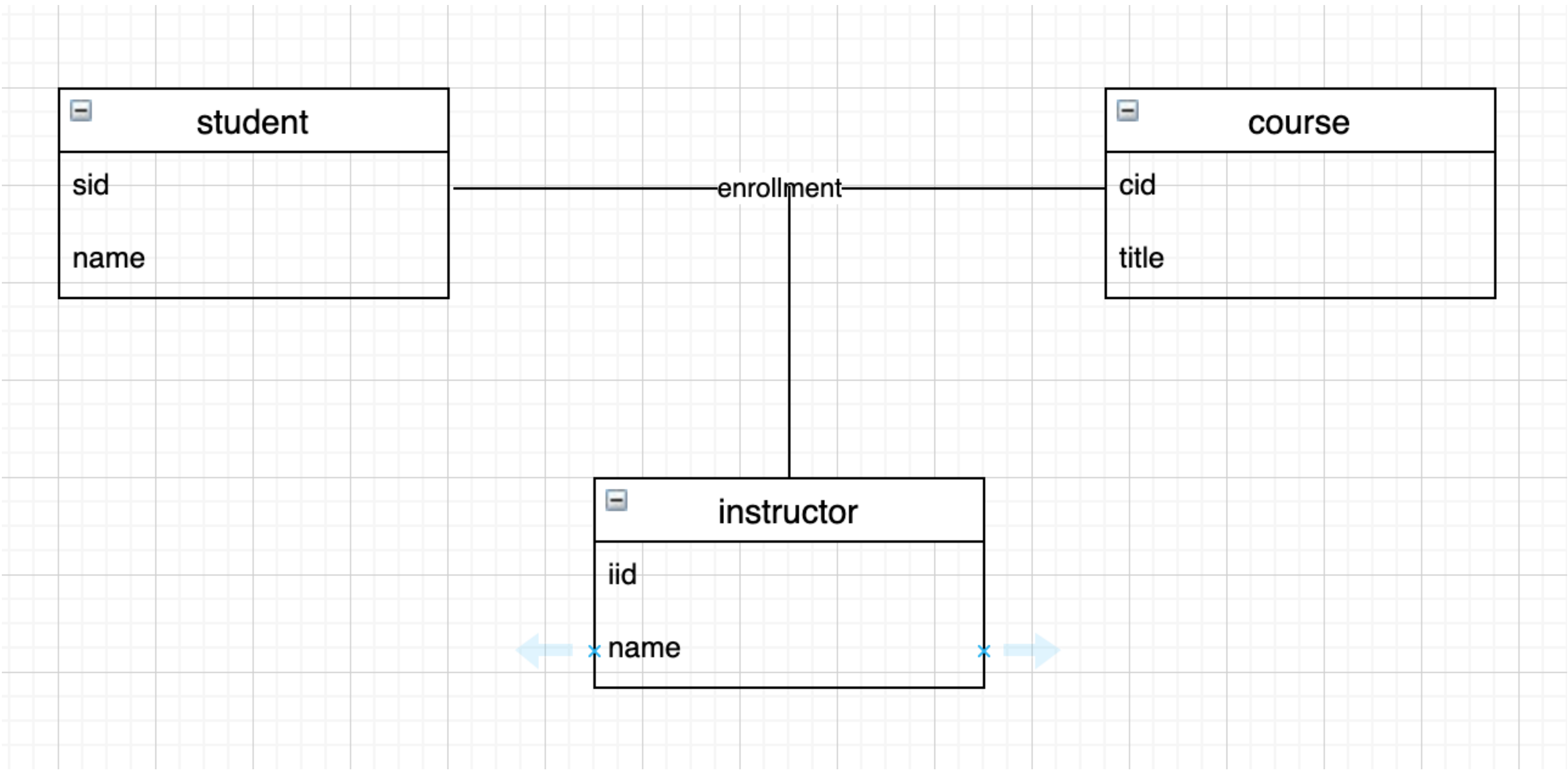
ER Modeling Example

- Requirements
  - Create a database for a course enrollment system in a university.
  - Students enroll into a courses a semester which are taught by instructors.
  - A course is taught by one and only one instructor

Entities & Relation

- Student
  - sid, name
- Course
  - cid, title
- Instructor
  - iid, name

First Conceptual Model



Limitations

- At the physical/implementation level, RDBMS doesn't support complex relations
  - without introducing overheads/redundancy
- Design principles play role here
  - while updating data, a minimal numbers of table should be involved
  - e.g., previous example
    - keep student/course/instructor info in the same table
      - same info will be repeated again/again
    - Now think about the updating a student?

Refining ERD

- Entities should participate in a relationship unless we have only one table in the database
- **one-to-one and one-to-many binary relationships** are good
  - – we don’t need to refine anything here

Refining ERD

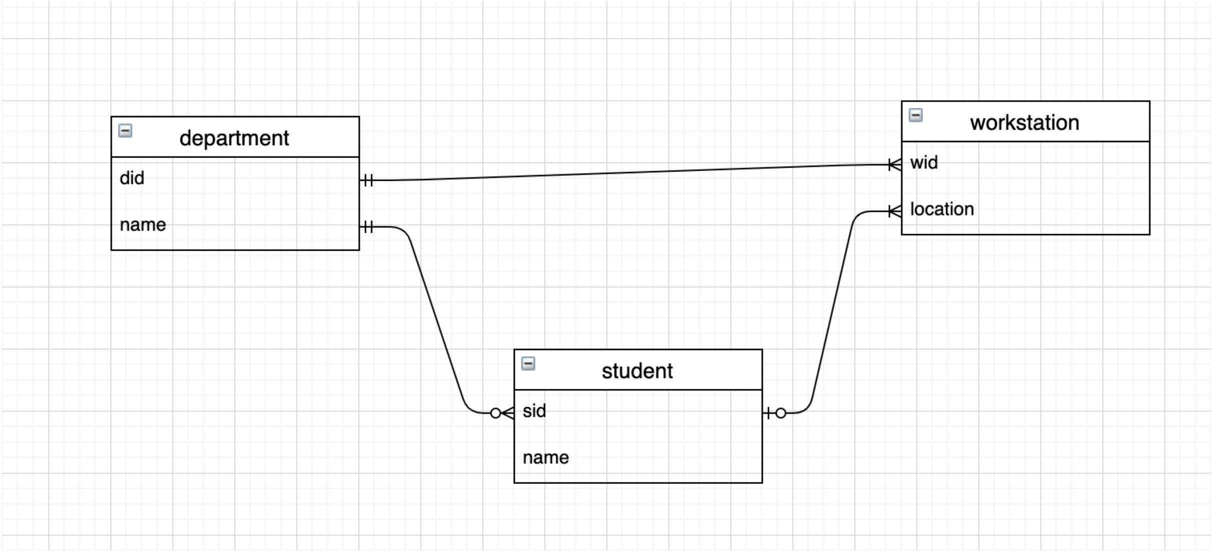
- **many-to-many binary relations** cannot be implemented without introducing redundancy
  - replace the relation with an intermediate entity
    - create the bridge/establish relationship
    - aka **association entity**
  - relate two original entities to this association entity

Refining ERD

- Complex relationship (ternary or n-ary)
  - No way database can implement this without overheads
  - Solution
    - Similar to **many-to-many binary relations**
    - Introduce an **association entity** for the complex relation
    - relate 3/n original entities to this association entity

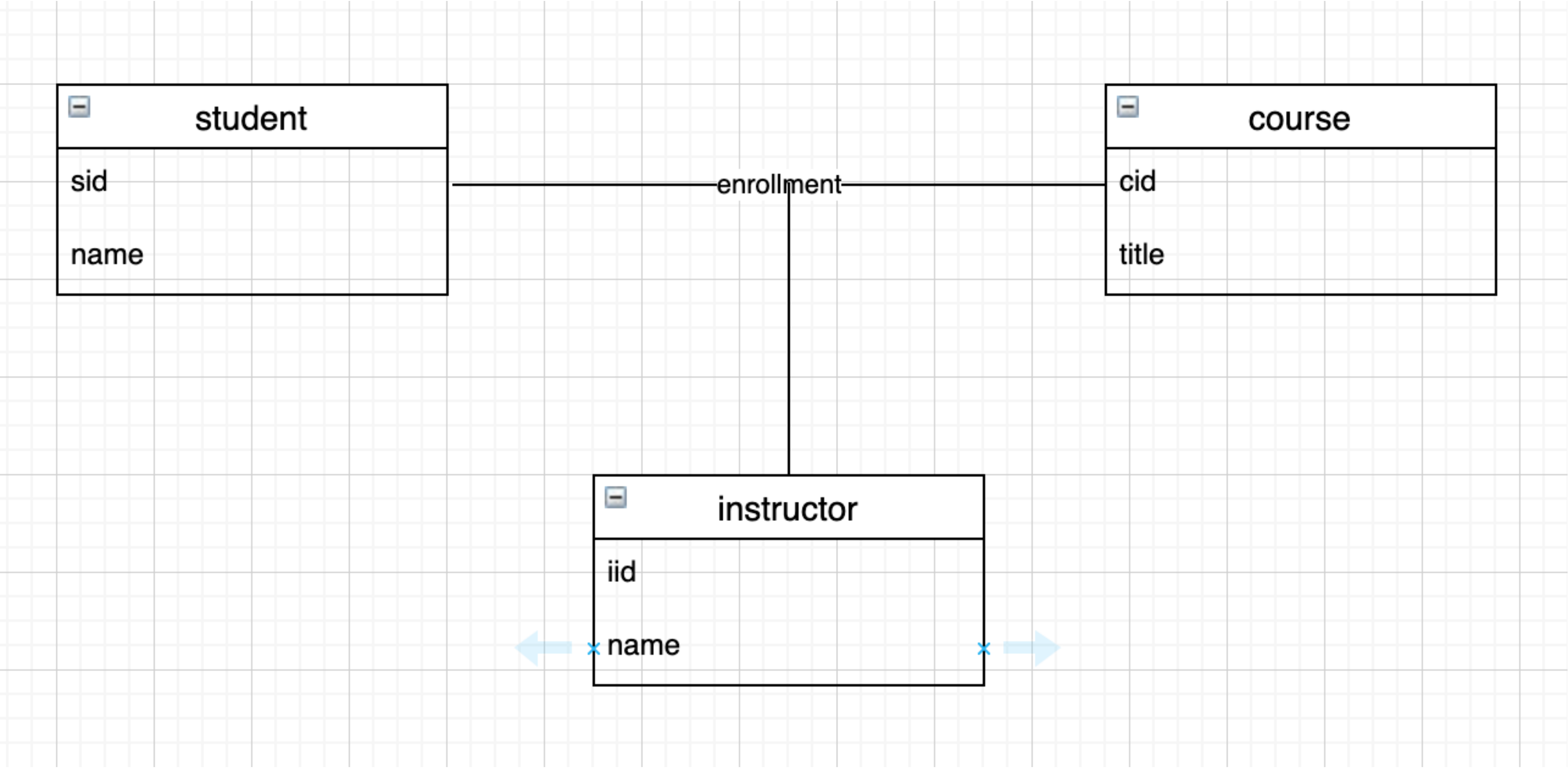
Refining ERD

- Eliminate redundant relations



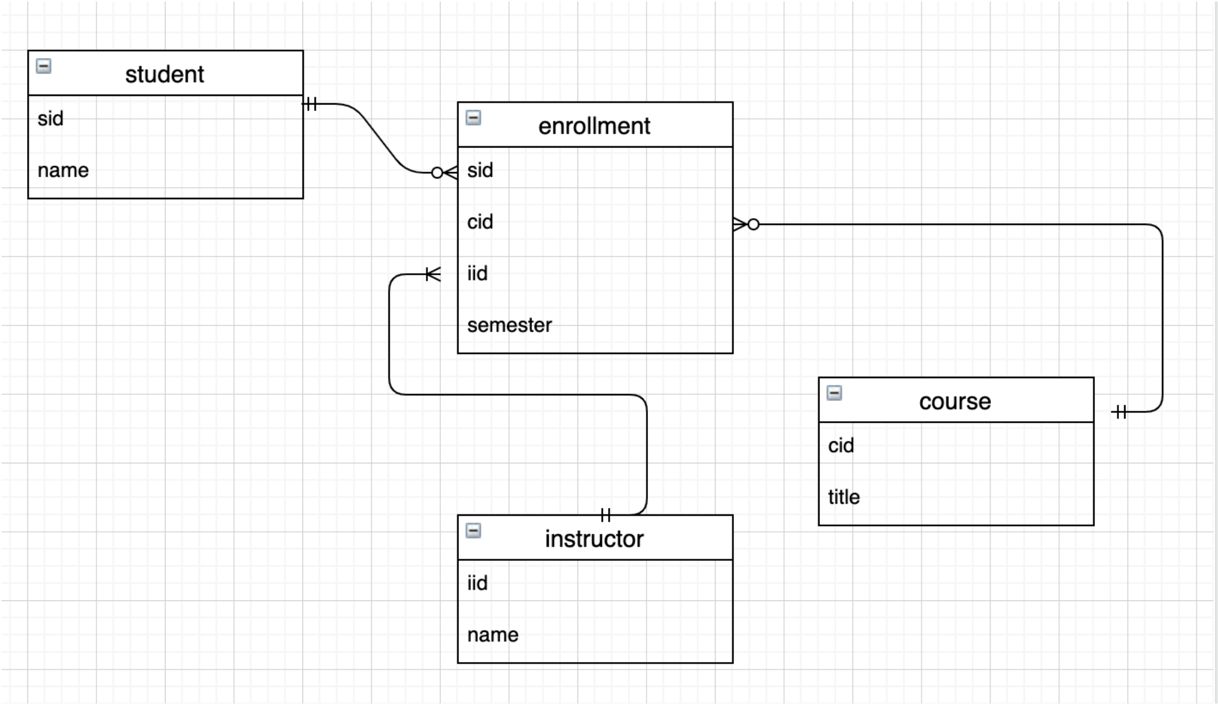
- Relation between department and workstation is redundant
  - Connection between department and workstation could be established via student table

Refining Course ERD



Ternary Relationship

Final Refinement



- Is the above solution good enough? Is there any other solutions? Yes.
  - Involves database normalization techniques

ERD to DDL

- Entities and Simple Attributes
  - An entity type within ER diagram is turned into a table
  - The key attribute of the entity is the **primary key** of the table which is usually underlined
    - Uniquely identify a row in the table
  - E.g. simple texual description of schema
    - student(sid,name)
    - cours(cid,title)

Converting ERD to DDL

- Entities and Simple Attributes
  - Convert to an SQL statement for creating the tables

```
CREATE TABLE STUDENT(
    sid char(20),
    name varchar(20)
);
```

- Note SQL sytax
  - case insensitive (unlike python/r)
  - typed (unlike python/r)
  - semicolon for ending a stmt (unlike python)

Using CHAR and VARCHAR

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Activity

- Switch to Jupyter notebook
- Create a dataset for this schema
- No need to fulfill key constraints

Destroying and Altering Relations

- DROP TABLE Student;
  - Destroy the relation student
    - schema is deleted
    - rows (i.e., tuple) are deleted
- ALTER TABLE Student

  ADD DOB date;
  - The schema of Student is altered by adding a new field
  - every tuple in the current instance is extended with a *null* value in the new field

Integrity Constraints (ICs)

- Condition that must be true for *any* instance of the database
  - e.g., domain constraint
    - The value of field came from a predefined set
- A **legal** instance of a relation is one that satisfies all specified ICs
  - DBMS should not allow illegal instances
- If the DBMS checks ICs, stored data is more faithful to real-world meaning

Where do ICs Come From?

- ICs are based upon the semantics of the real-world enterprise
  - Discover during requirement analysis
- Key and foreign key ICs are the most common

Primary Key Constraints

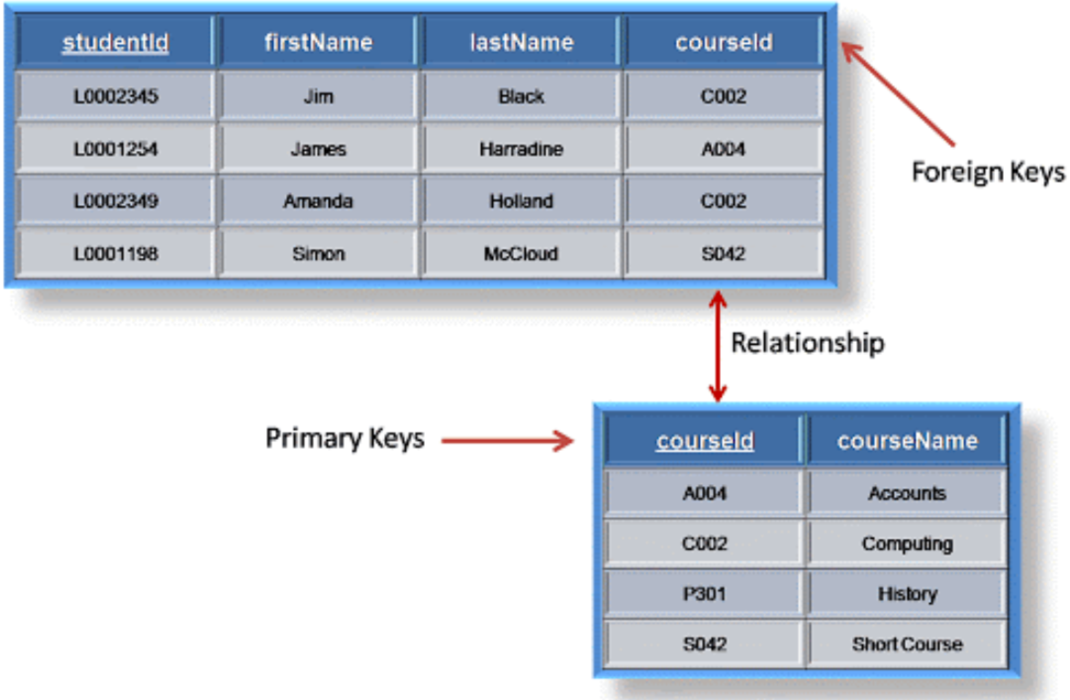
- A set of fields is a *key* for a relation if :
  - No two distinct tuples can have same values in all key fields, and
  - This is not true for any subset of the key. (minimal set of attributes)
- E.g., *sid* is a key for Students.
  - What about *name*?
  - What about (sid, name)?
  - The set {*sid, gpa*} is a superkey.

Primary Key Constraints

```
CREATE TABLE Student (
        sid CHAR(20),
        name VARCHAR(20),
         PRIMARY KEY(sid)
);
```

Foreign Keys, Referential Integrity

- Set of fields in one relation that is used to `refer' to a tuple in another relation
  - Must correspond to primary key of the second relation



studentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

courseId	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course

Foreign Keys, Referential Integrity

- E.g. *sid* is a foreign key in enrollment table referring to Students
- *Foreign Keys in SQL*

```
CREATE TABLE Enrollment (
        sid CHAR(20),
        cid CHAR(20),
        iid CHAR(20),
        PRIMARY KEY (sid,cid,iid),
        FOREIGN KEY (sid) REFERENCES Student (sid)
        ….
);
```