CS 327E Class 2

Jan 29, 2021

Relational Data Model

- Database == Collection of relations
- Relation == A table with columns (attributes) and rows (tuples)
- Column properties: named, domain, unordered
- Row properties: single-valued attributes, unique, unordered

How do we enforce a unique row constraint?

- Referential integrity: Every non-null foreign key must match an existing primary key value.

Notation: Customer(id, fname, lname, address ...)

Order(orderno, *custid*, date, channel ...)

SQL Queries: CRUD Operations

```
SELECT c1, c2, c3, cn
FROM T1
WHERE c1 > 100 OR c1 < 200
ORDER BY c3, c4;

SELECT c1, c2, c3, cn
FROM T1
WHERE c1 IS NOT NULL
ORDER BY c3 DESC;
```

More CRUD Operations

```
CREATE TABLE T1 (c1 INT PRIMARY KEY,
                 c2 VARCHAR(30) NOT NULL,
                 c3 VARCHAR(30));

INSERT INTO T1 (c1, c2, c3) VALUES (1, 'Austin',
 'TX');

UPDATE T1 SET c2 = 'New York City', c3 = 'NY'
 WHERE c1 = 1;

DELETE FROM T1 WHERE c3 IN ('NY' 'TX', 'CA');
```

Why MySQL?

- It’s been around a long time
- Simple and easy-to-use
- Open-source software
- Implements the relational model
- Designed for storing structured data
- Feature-rich SQL support
- Supports many languages
- Small to medium size data (< TB storage)
- Low to moderate QPS of reads and writes (10K)
- Read replicas for scaling reads
- Sharding for scaling writes (e.g. Vitess)

Instapoll on today’s set up

MySQL Guide:
https://github.com/cs327e-spring2021/snippets/wiki/MySQL-Setup-Guide

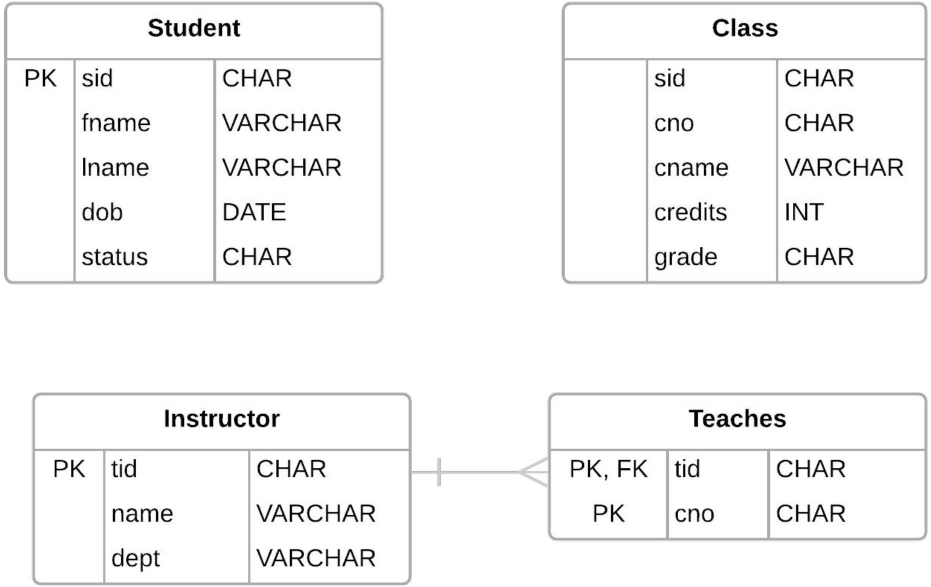
Jupyter Guide:

https://github.com/cs327e-spring2021/snippets/wiki/Jupyter-Setup-Guide

Let’s start working with MySQL:

- Clone snippets repo
- Open mysql notebook
- Create database
- Create tables
- Populate tables
- Check tables
- Remove header row
- Add primary keys
- Add foreign key
- Test foreign key

College Database Schema



Student

PK	sid	CHAR
	fname	VARCHAR
	lname	VARCHAR
	dob	DATE
	status	CHAR

Class

	sid	CHAR
	cno	CHAR
	cname	VARCHAR
	credits	INT
	grade	CHAR

Instructor

PK	tid	CHAR
	name	VARCHAR
	dept	VARCHAR

Teaches

PK, FK	tid	CHAR
PK	cno	CHAR

Student(sid, fname, lname, dob, status)

Class(sid, cno, cname, credits, grade)

Instructor(tid, name, dept)

Teaches(*tid*, cno)

Practice Problems

Who takes CS327E *or* CS329E?

Who takes CS327E *and* CS329E?

Student(sid, fname, lname, dob, status)

Class(sid, cno, cname, credits, grade)

Instructor(tid, name, dept)

Teaches(*tid*, cno)

Second Question

Student(sid, fname, lname, dob, status)

Class(cno, cname, credits)

Instructor(tid, name, dept)

Teaches(*tid*, cno)

Who takes CS327E *and* CS329E?

Is this query a correct implementation?

```
SELECT sid
FROM Current_Student
WHERE cno = 'CS327E'
  AND cno = 'CS329E'
```

Relational Data Modeling

- Entity: A real-world object
- Usually a noun
- Common examples: Person, Team, Product, Order, Shipment

Analogies with OOP:

- Entity: analogous to class
- Record: analogous to objects
- Attribute: analogous to members of an object

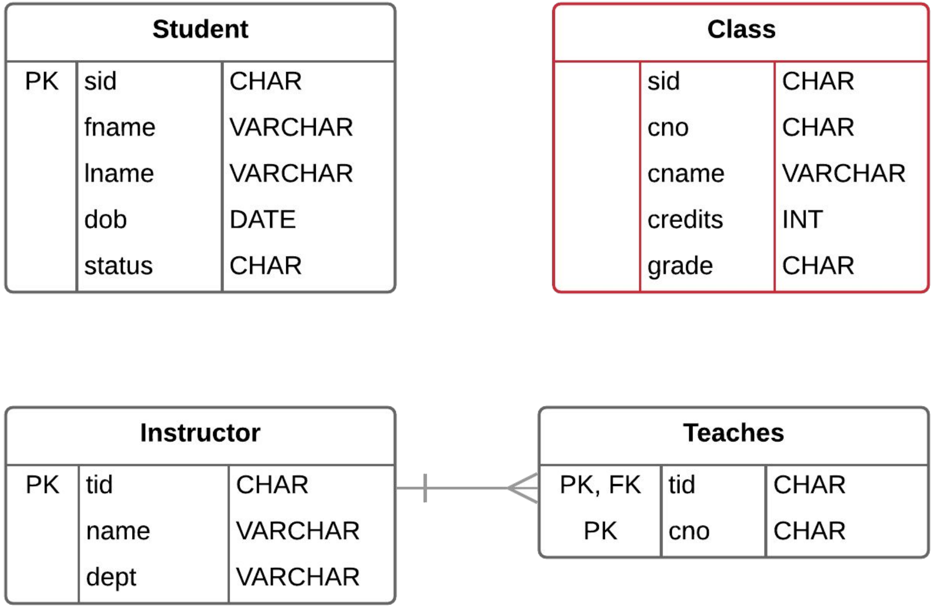
Questions:

- How do we represent relationships between entities?
- Can entities have methods in addition to members?

Design Guidelines

1. A table models a single entity and an entity is modeled by a single table.
2. The collection of fields of an entity represent the attributes of that entity.
3. Each field is given a primitive type that best fits its domain of values.
4. Each table has a primary key (PK) which is made up of one or more fields that uniquely represent each record.
5. A child table has a foreign key (FK) which references its parent’s PK.
6. A *m:n* relationship is modeled as a junction table.

Back to our college schema:



Student

PK	sid	CHAR
	fname	VARCHAR
	lname	VARCHAR
	dob	DATE
	status	CHAR

Class

	sid	CHAR
	cno	CHAR
	cname	VARCHAR
	credits	INT
	grade	CHAR

Instructor

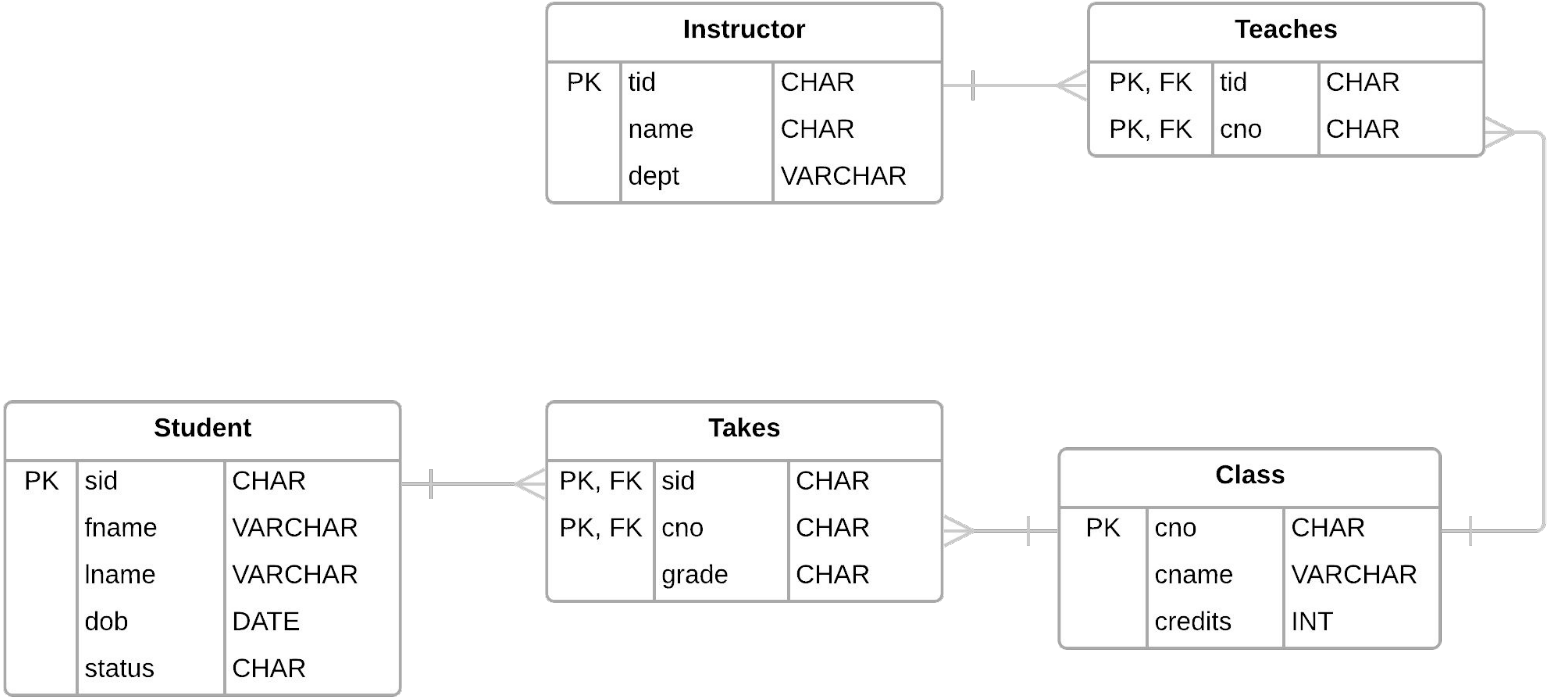
PK	tid	CHAR
	name	VARCHAR
	dept	VARCHAR

Teaches

PK, FK	tid	CHAR
PK	cno	CHAR

- Insert Anomaly
- Update Anomaly
- Delete Anomaly

Remodeled college schema



Instructor

PK	tid	CHAR
	name	CHAR
	dept	VARCHAR

Teaches

PK, FK	tid	CHAR
PK, FK	cno	CHAR

Student

PK	sid	CHAR
	fname	VARCHAR
	lname	VARCHAR
	dob	DATE
	status	CHAR

Takes

PK, FK	sid	CHAR
PK, FK	cno	CHAR
	grade	CHAR

Class

PK	cno	CHAR
	cname	VARCHAR
	credits	INT

Common Transforms

-
  ```
  CREATE TABLE T2 AS SELECT a, b, c FROM T1
  ```

-
  ```
  SELECT a, b, c FROM T1
  UNION [DISTINCT]
  SELECT x AS a, y AS b, z AS c FROM T2
  ```

-
  ```
  SELECT a, b, c, 'some string' AS s FROM T1
  UNION ALL
  SELECT d, e, f, 'some string' AS s FROM T2
  ```

Project 1

http://www.cs.utexas.edu/~scohen/projects/Project1.pdf