The Relational Model
ER to Relational / SQL

Announcements

Project 1 Part 1: Starting tomorrow!
Partners: Try Piazza

Review

Relation: Set of tuples with typed values (table)
Schema: Names and types for values in relation
Database: Set of relations
SQL: Structured Query Language
Integrity constraint: Restrictions on valid data
Candidate key: Minimal fields that identify tuple
Primary key: Designated identifier for a tuple
Foreign key: Reference to another tuple

Referential Integrity

A database instance has *referential integrity* if all foreign key constraints are enforced no dangling references

Examples where referential integrity is not enforced
- Web links
- Restaurant menus
- Some relational databases!

How to Enforce Integrity Constraints

Run checks any time database changes

On INSERT
- what if new Enrolled tuple refers to non-existent student? Reject insertion

On DELETE (many options)
- what if Students tuple is deleted?
  - delete dependent Enrolled tuples
  - reject deletion
  - set Enrolled.sid to default value or null (null in SQL: unknown/inapplicable)

How to Enforce Integrity Constraints

Run checks any time database changes

On INSERT
- what if new Enrolled tuple refers to non-existent student? Reject insertion

On DELETE (many options)
- what if Students tuple is deleted?
  - delete dependent Enrolled tuples (CASCADE)
  - reject deletion (default: ON DELETE NO ACTION)
  - set Enrolled.sid to default value or null (null in SQL: unknown/inapplicable)

General Constraints

Boolean constraint expression added to schema
Very powerful; not common

```
CREATE TABLE Enrolled(
    sid int,
    cid int,
    grade char(2),
    CHECK (
        boolean expression
    )
)
```

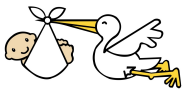
General Constraints

Boolean constraint expression added to schema
Very powerful; not common

```
CREATE TABLE Enrolled(
    sid int,
    cid int,
    grade char(2),
    CHECK (
        grade = 'A' or grade = 'B' or
        grade = 'C' or grade = 'D' or
        grade = 'F'
    )
)
```

Where do ICs come from?

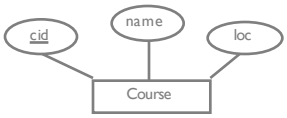
Based on application semantics and use cases
IC is statement about the world (all possible instances)
Can't infer ICs by staring at an instance
e.g. Is "login" a unique candidate key? Maybe?
Unique and foreign key ICs are very common
(others less so)



Translating ER → Relational Models

Entity Set → Relation

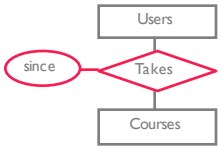
Include all attributes
Entity set key become relation primary key




```
CREATE TABLE Course(
    cid int,
    name text,
    loc text,
    PRIMARY KEY (cid)
)
```

Relationship Set w/out constraint → Relation

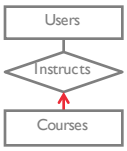
Add keys for each entity set as foreign keys: *superkey*
(if there are other constraints: may not need all columns)
All attributes of the relationship set




```
CREATE TABLE Takes(
    uid int,
    cid int,
    since date,
    PRIMARY KEY (uid, cid),
    FOREIGN KEY (uid) REFERENCES Users,
    FOREIGN KEY (cid) REFERENCES Courses
)
```

At Most One → Relation

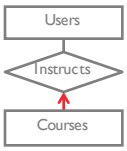
Add relationship attributes (none here)
Add keys for entity set as foreign keys
What is the primary key?




```
CREATE  TABLE  Instructs(
    uid int,
    cid int,
    PRIMARY KEY (cid),
    FOREIGN  KEY (uid)  REFERENCES  Users,
    FOREIGN  KEY (cid)  REFERENCES  Courses
)
```

At Most One: Combine?

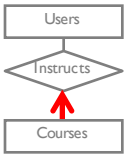
Zero or 1 courses, cid is primary key; Similar to ???
Combine Instructs attributes into Courses (preferred)
How to represent courses without instructor?
NULL uid (and other Instructs attributes)




```
CREATE  TABLE  Course_Instructs (
    cid int
    uid int,  Maybe rename?
    name  text,
    loc  text,
    PRIMARY  KEY (cid),
    FOREIGN  KEY (uid)  REFERENCES  Users
)
```

Exactly One Constraint → Relation

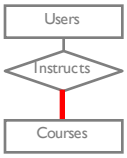
Represent: Course must have Instructor?
Combine relationship into Courses + NOT NULL
What happens if we delete User who is Instructor?
Default ON DELETE NO ACTION: Don't permit




```
CREATE  TABLE  Course_Instructs (
    cid int
    uid int, NOT NULL,
    name  text,
    loc text,
    PRIMARY  KEY (cid),
    FOREIGN  KEY (uid)  REFERENCES  Users
)     ON DELETE  NO ACTION
)
```

At Least One Constraint?

```
CREATE TABLE Instructs(
   cid int,
   uid int,
   PRIMARY KEY (cid, uid),
   FOREIGN KEY (cid) REFERENCES Course,
   FOREIGN KEY (uid) REFERENCES User)
```

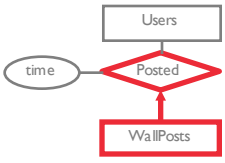


Can't be easily expressed!

Weak Entity → Relation

Weak entity set and identifying relationship set are translated into a single table.

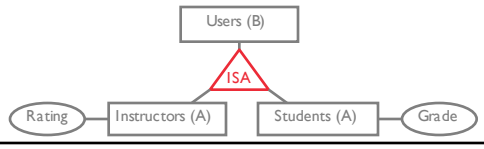
When the owner entity is deleted, all owned weak entities must also be deleted.




```
CREATE TABLE Wall_Posted(
    uid int,
    post_text text,
    posted_time DATE
    PRIMARY KEY (uid, posted_time),
    FOREIGN KEY (uid) REFERENCES Users
        ON DELETE CASCADE
)
```

ISA Hierarchies

```
CREATE TABLE Users(uid int, name text, PRIMARY KEY(uid))
CREATE TABLE Instructors(uid int, rating int,
    PRIMARY KEY(uid), FOREIGN KEY (uid) REFERENCES Users)
CREATE TABLE Students(uid int, grade char(2),
    PRIMARY KEY(uid), FOREIGN KEY (uid) REFERENCES Users)
```

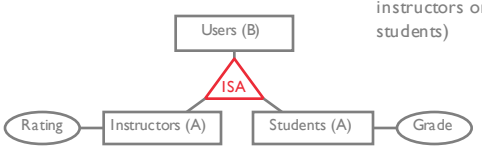


ISA Hierarchies

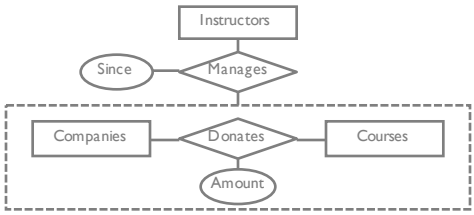
Option 1: Keep base relation
- Instructors & Students recorded in Users
- Extra info in Instructors or Students relation
- Primary key = key of parent table; is foreign key for child tables

Option 2: Only keep child relations
- Children copy attributes from Users

Only if covering constraint = yes (all users are instructors or students)

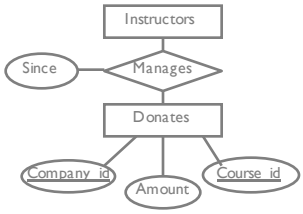


Aggregation



Aggregation

Convert the aggregated relationship into an entity
Convert like any other entity
E.g. Donates: PRIMARY KEY (Company_id, Course_id)
Manages: References this key



ER-Relational Review

Not complete. Just tips

The textbook is actually quite good on this topic!

Postgres documentation

Good reference on SQL

https://www.postgresql.org/docs/9.3/static/ddl.html

Feedback

http://tinyurl.com/w4111