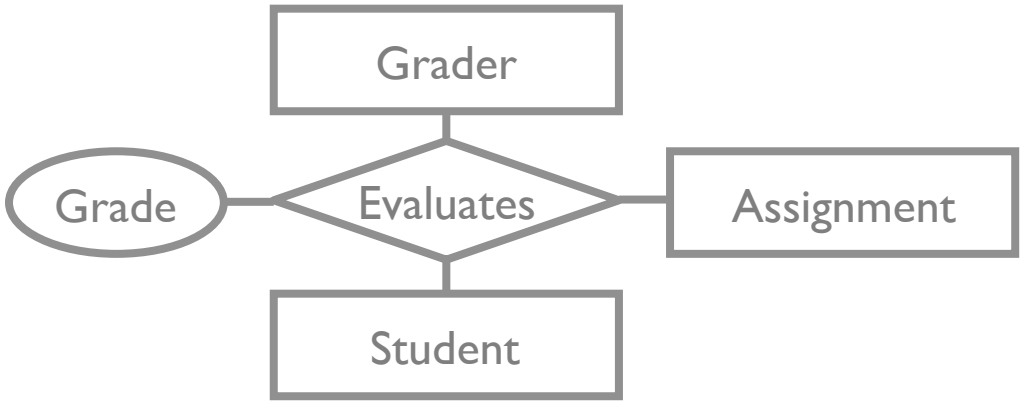
Relational Model

Announcements

- Waiting list: we are done and full with 110 students
- Project 1: You should have a team and should have signed up for an advising section

Ternary relationship and constraints



Grader
Jane
Neha
Lin
…

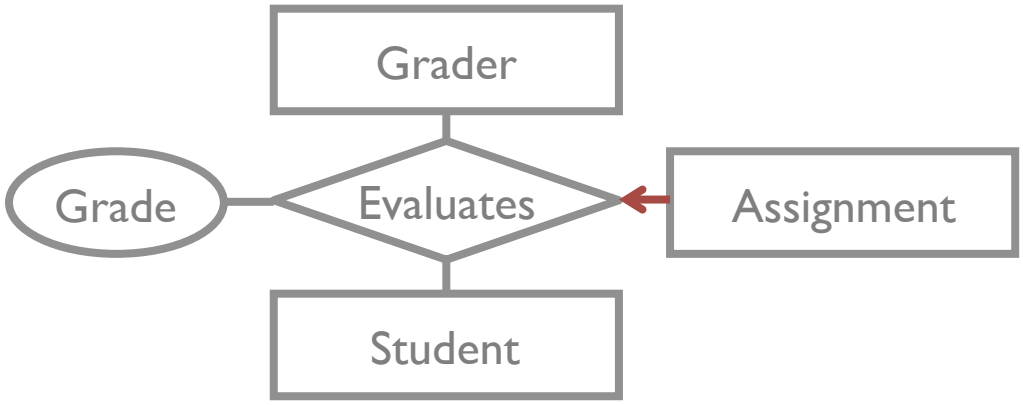
Student
Alice
Jinyang
Sarah
…

Assignment
Homework 0
Project 1
…

Assignment	Student	Grader	Grade
Homework 0	Jinyang	Jane	8
Homework 0	Alice	Jane	7
Homework 1	Jinyang	Neha	7
Homework 0	Alice	Lin	8
Homework 0	Sarah	Neha	6

Part of class syllabus, *not* a specific submission

Ternary relationship and constraints

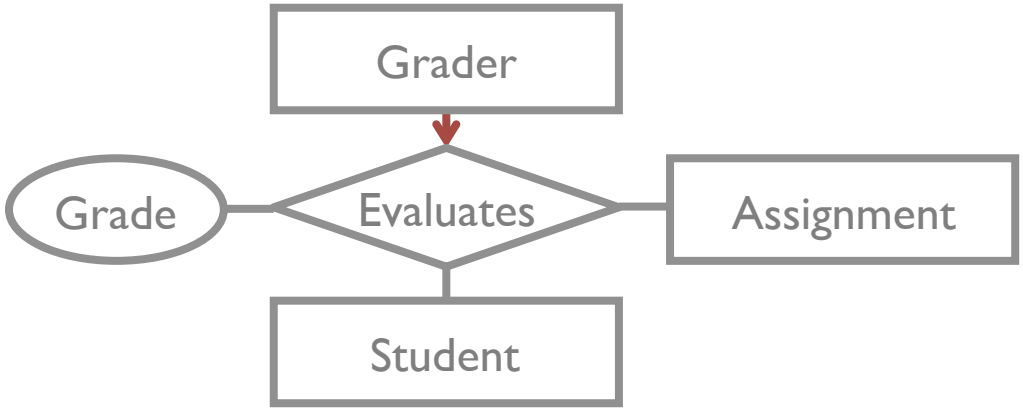


At most one grader per assignment?

HW0 can appear at most once! Also at most one student

Assignment	Student	Grader	Grade
Homework 0	Jinyang	Jane	8
~~Homework 0~~	~~Alice~~	~~Jane~~	~~7~~
Homework 1	Jinyang	Neha	7
~~Homework 0~~	~~Alice~~	~~Lin~~	~~8~~
~~Homework 0~~	~~Sarah~~	~~Neha~~	~~6~~

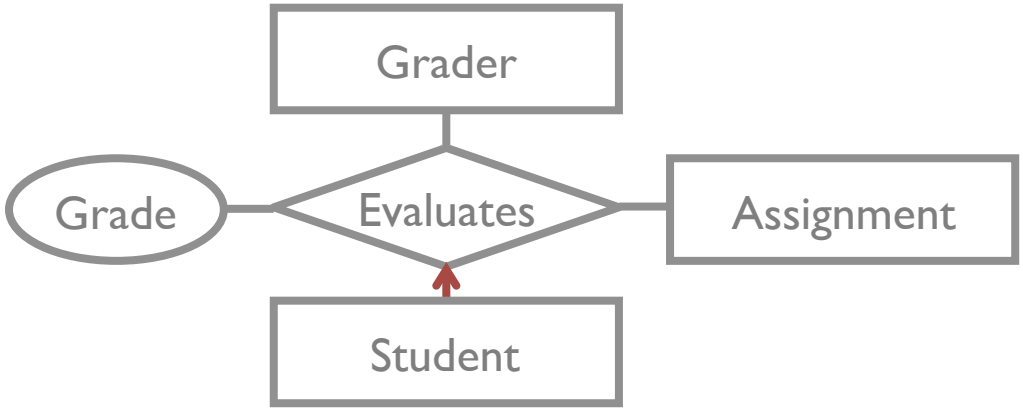
Ternary relationship and constraints



At most one grader per assignment?

Assignment	Student	Grader	Grade
Homework 0	Jinyang	Jane	8
~~Homework 0~~	~~Alice~~	~~Jane~~	~~7~~
Homework 1	Jinyang	Neha	7
Homework 0	Alice	Lin	8
~~Homework 0~~	~~Sarah~~	~~Neha~~	~~6~~

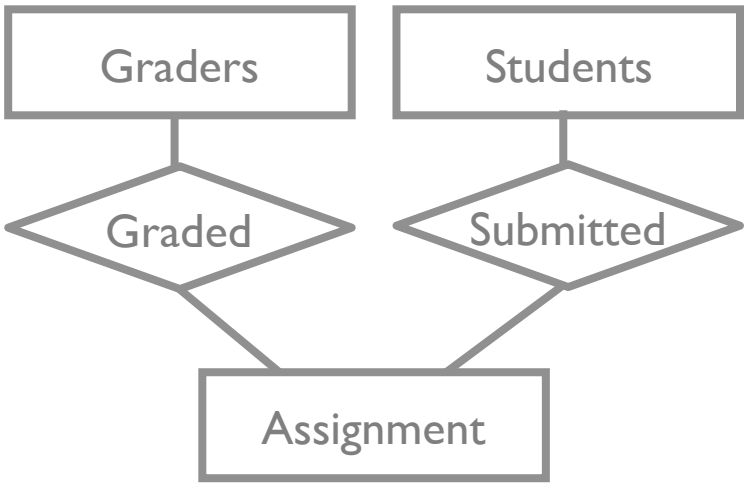
Ternary relationship and constraints



At most one grader per assignment?

Assignment	Student	Grader	Grade
Homework 0	Jinyang	Jane	8
Homework 0	Alice	Jane	7
~~Homework 1~~	~~Jinyang~~	~~Neha~~	~~7~~
~~Homework 0~~	~~Alice~~	~~Lin~~	~~8~~
Homework 0	Sarah	Neha	6

Ternary relationship and constraints

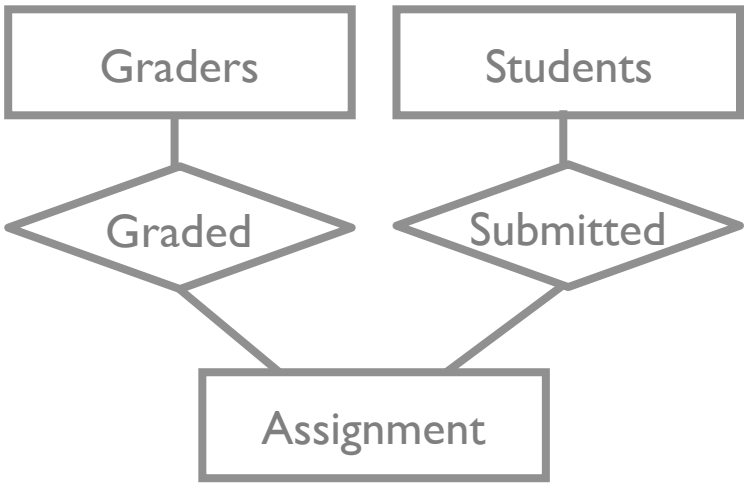


Assignment	Grader
Homework 0	Jane
Homework 1	Neha
Homework 0	Lin
Homework 0	Neha

Assignment	Student	Grader	Grade
Homework 0	Jinyang	Jane	8
Homework 0	Alice	Jane	7
Homework 1	Jinyang	Neha	7
Homework 0	Alice	Lin	8
Homework 0	Sarah	Neha	6

Assignment	Student
Homework 0	Jinyang
Homework 0	Alice
Homework 1	Jinyang
Homework 0	Sarah

Ternary relationship and constraints

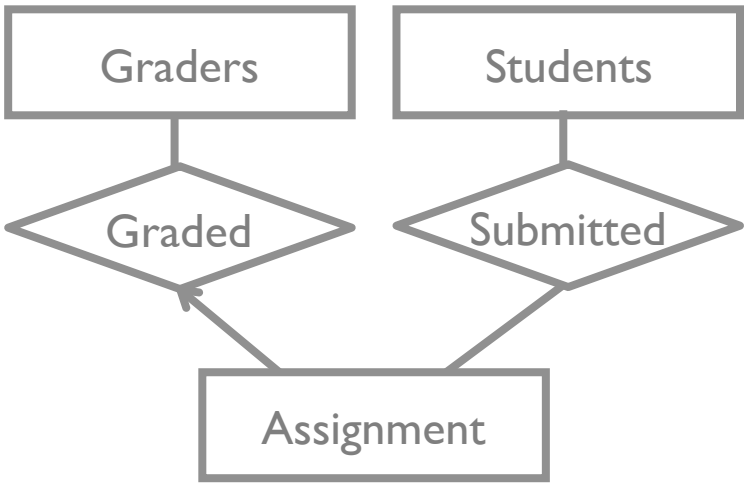


Assignment	Grader
Homework 0	Jane
Homework 1	Neha
Homework 0	Lin
Homework 0	Neha

At most one grader per assignment?

Assignment	Student
Homework 0	Jinyang
Homework 0	Alice
Homework 1	Jinyang
Homework 0	Sarah

Ternary relationship and constraints



Assignment	Grader
Homework 0	Jane
Homework 1	Neha
~~Homework 0~~	~~Lin~~
~~Homework 0~~	~~Neha~~

Assignment	Student
Homework 0	Jinyang
Homework 0	Alice
Homework 1	Jinyang
Homework 0	Sarah

At most one grader per assignment?

Students can only submit a given assignment once?

Enforced: relationships are sets

No weak entities here

Roadmap

- History lesson
- DDLs: Data definition language
- Integrity Constraints
- DMLs: Data Manipulation Language Selection Queries
- ER → Relational Model

Relational History

- 70s Big debate: network vs relational model
- IBM: IMS powered all “real” apps on mainframes
- Oracle, Ingres: DBs for minicomputers (VAX)
- 1984: IBM DB/2 with SQL for mainframes
- Killed other models and languages

- Still a huge industry: Oracle, IBM, Microsoft,
- HP Vertica, Teradata, others

Basic Definitions

- Database a set of relations

- Relation a table with rows and columns
  - Schema name of relation + name & type of each column
  - Instance specific set of rows
  - e.g., Students(sid: int, name: string, login: string, age: int)

- Think of relation as a *set* (no duplicate rows)
- Relation colored glasses
  - Everything (data, relationships, query results) is a relation

Terminology

Formal Name	Synonyms
Relation	Table
Tuple	Row, Record
Attribute	Column, Field
Domain	Type
Cardinality	# of tuple
Degree	# of attributes

Example *Instance* of Students Relation

sid	name	login	age	gpa
1	eugene	ewu@cs	20	2.5
2	neha	neha@cs	20	3.5
3	lin	lin@math	33	3.9

- Cardinality 3
- Degree 5

- Do rows have to be distinct? (Yes)
- Do columns have to be distinct? (No)

Integrity Constraints (ICs)

- def: a condition that is true for *any* instance of the database

- Often specified when defining schema
- DBMS enforces ICs at all times

- An instance of a relation is legal if it satisfies all declared ICs
- Programmer doesn't have to worry about data errors!
- e.g., data entry errors

- Don't Repeat Yourself (DRY)
- PostgreSQL documentation great resource
- www.postgresql.org/docs/8.1/static/ddl-constraints.html

SQL DDL: CREATE TABLE

- `CREATE TABLE` *`Name`*`(`
- *`columnName columnType`*`,`
- …
- `)`

Domain Constraints (attr types)

```
CREATE TABLE Students(
   sid int,
   name text,
   login text,
   age int,
   gpa real
)
```

SQL DDL: CREATE TABLE

- Create the Students Relation

```
CREATE TABLE Students(
    sid int,
    name text,
    login text,
    age int,
    gpa real
)
```

- Note: attribute domains are defined & enforced by DBMS

Create the Enrolled relation

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

Adding data

- INSERT INTO Students VALUES
- (1, “Evan”, “ej”, 34, 3.1),
- (2, “Jinyang”, “jinyang”, 18, 3.9);

NULL Constraints

Default: Columns can contain the special value NULL (no value, optional)

Exception: Primary keys

```
CREATE TABLE Students(
   sid: int NOT NULL,
   name: text,
   login: text,
   age: int,
   gpa: float
)
```

Candidate Keys

- Set of fields is a *candidate key (or just Key)* for a relation if:

1. Two distinct valid tuples cannot have same values
2. This is **not** true for any subset of the key (minimal)

- If (2) is false, called a superkey what's a trivial superkey?

- If >1 candidate keys in relation, admin assigns primary key:
- Used to identify tuples elsewhere in the database

- sid is key for Students
- is name a key?
- what is (sid, gpa)?

Primary and Candidate Keys

- UNIQUE & PRIMARY KEY key words
- Be careful with integrity constraints:

Each student can enroll in a course only once

```
CREATE TABLE Enrolled(
    sid: int,
    cid: int,
    grade: char(2),
    PRIMARY KEY (sid, cid)
)
```

What does this say?

```
CREATE TABLE Enrolled(
    sid: int,
    cid: int,
    grade: char(2),
    PRIMARY KEY (sid),
    UNIQUE (cid, grade)
)
```

Foreign Keys

- def: set of fields in Relation R_i used to refer to
- tuple in R_j via R_j's primary key (logical pointer)

```
CREATE TABLE Enrolled(
    sid: int,  cid: int,  grade: char(2),
    PRIMARY KEY (sid, cid),
    FOREIGN KEY (sid) REFERENCES Students
)
```

Enrolled

sid	cid	grade
1	2	A
1	3	B
2	2	A+

Students

sid	name
1	eugene
2	luis

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
- HTML links
- Yellow page listing
- Restaurant menus
- Some relational databases!

How to Enforce Integrity Constraints

- Run checks anytime 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 means 'unknown' or 'inapplicable' in SQL)

General Constraints

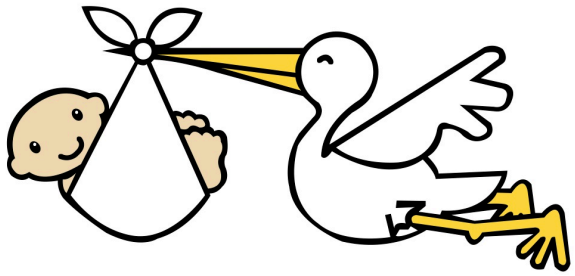
- Boolean constraint expression added to schema
- Very powerful mechanism.
- More specific constraints in next slides

```
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
- Can check if database instance satisfies ICs
- IC is statement about the world (all possible instances)
- Can't infer ICs by staring at an instance

- Key and foreign key ICs are most common, more general table aints possible as well.



More Powerful than ER Constraints

- Functional dependencies
  - *A dept can't order two distinct parts from the same supplier.*
  - Cant express this with ternary Contracts relationship.
  - Normalization refines ER design by considering FDs.
- Inclusion dependencies
  - Special case: ER model can express Foreign keys
  - *At least 1 person must report to each manager.*
- General constraints
  - *Each donation is less than 10% of the combined donations to all humanities courses.*

What Can ER Express?

- Key constraints, participation constraints, overlap/ covering constraints

- Some foreign key constraints as part of relationship set
- Some constraints require general CHECK stmts
- ER cannot express e.g., function dependencies at all
- Constraints help determine best database design

DML: Introduction to Queries

- Key strength of relational model
- declarative querying of data

- Queries are high level, “readable”
- DBMS makes it fast, user don’t need to worry

- Precise semantics for relational queries
  - Lets DBMS choose different ways to run query while ensuring answer is the same

INSERT/DELETE

- Add a tuple
  - `INSERT INTO Students(sid, name, login, age, gpa)`
  - `VALUES (4, ‘wu’, ‘wu@cs’, 20, 5)`

- Delete tuples satisfying a predicate (condition)

```
DELETE FROM Students S
WHERE S.name = ‘wu’
```

Basic SELECT

- Get all attributes of <21 year old students

```
SELECT *
FROM Students
WHERE Students.age < 21
```

- Get only names

```
SELECT S.name
FROM Students S
WHERE S.age < 21
```

sid	name	login	age	gpa
1	eugene	ewu@cs	20	2.5
2	luis	luis@cs	20	3.5
3	ken	ken@math	33	3.9

Multi-table SELECT

- What does this return?

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid = E.sid AND
      E.grade = "A"
```

Enrolled

sid	cid	grade
1	2	A
1	3	B
2	2	A+

Students

sid	name
1	eugene
2	luis

Result

name	cid
eugene	2

Single Table Semantics

- A *conceptual evaluation method* for previous query:
  - 1. FROM clause: retrieve Students relation
  - 2. WHERE clause: Check conditions, discard tuples that fail
  - 3. SELECT clause: Delete unwanted fields

- Remember, this is *conceptual*. Actual evaluation will be *much* more efficient, but must produce the same answers.

Multi-Table Semantics

- Modify the FROM clause evaluation
  - 1. FROM clause: compute *cross-product* of Students and Enrolled

Enrolled

sid	cid	grade
1	2	A
1	3	B
2	2	A+

Students

sid	name
1	eugene
2	luis

Cross-product

sid	cid	grade	sid	name
1	2	A	1	eugene
1	3	B	1	eugene
2	2	A+	1	eugene
1	2	A	2	luis
1	3	B	2	luis
2	2	A+	2	luis

Multi-Table Semantics

- Modify the FROM clause evaluation
  - 1. FROM clause: compute *cross-product* of Students and Enrolled
  - 2. WHERE clause: Check conditions, discard tuples that fail
  - 3. SELECT clause: Delete unwanted fields