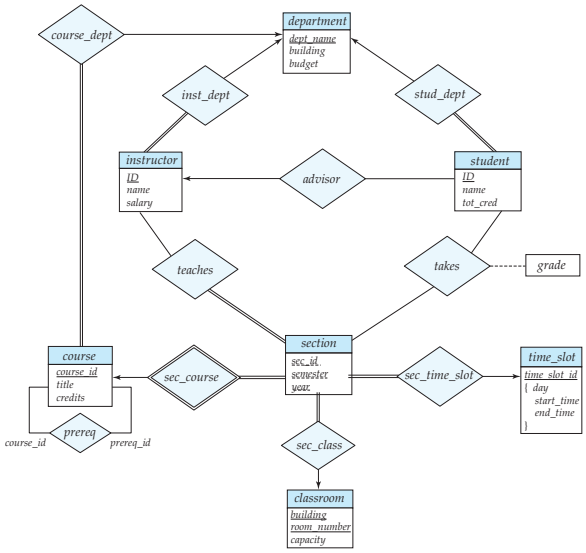
Outline

- Relational Algebra (6.1)
- E/R Model (7.2 - 7.4)
- E/R Diagrams (7.5)
- Reduction to Schema (7.6)
- Relational Database Design (7.7)
- Functional Dependencies (8.1 – 8.4)
- Normalization (8.5 – 8.7)
- Relational Query Languages
- SQL Basics
- Formal Semantics of SQL

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



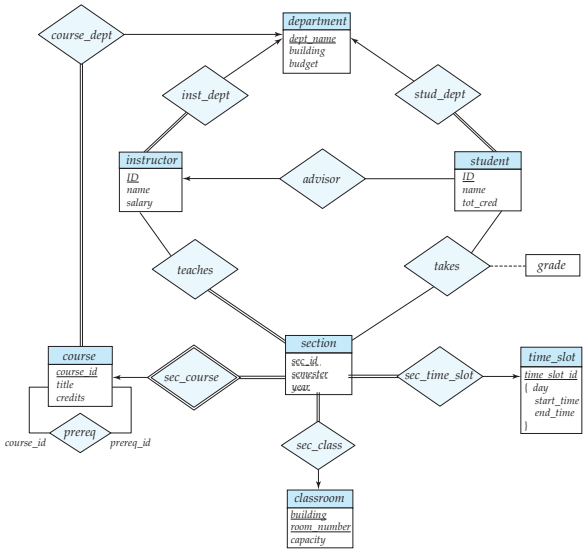
- department(dept_name, building, budget)
- instructor(ID, name, salary)
- course(course_id, title, credits)
- section
- student(ID, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, {(day, start_time, end_time)}

- inst_dept
- stud_dept
- teaches
- takes
- advisor(i_id, s_id)
- course_dept(course_id, dept_name)
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- sec_course(*mess*)
- prereq(course_id, prereq_id)
- sec_class(course_id, sec_id, semester, year, building, room_num)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



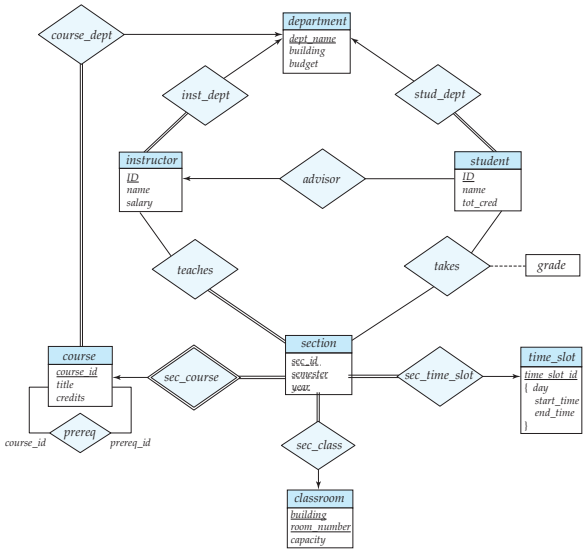
- department(dept_name, building, budget)
- instructor(ID, name, salary)
- course(course_id, title, credits)
- section(course_id, sec_id, semester, year)
- student(ID, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- inst_dept(ID, dept_name)
- stud_dept(ID, dept_name)
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- course_dept(course_id, dept_name)
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- sec_course(*mess*)
- prereq(course_id, prereq_id)
- sec_class(course_id, sec_id, semester, year, building, room_num)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



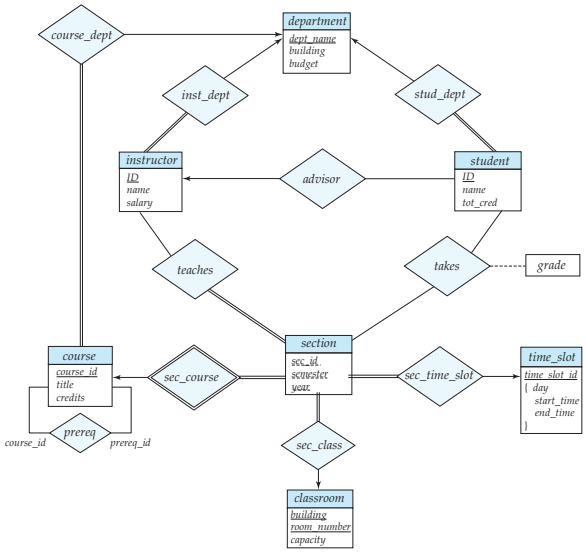
- department(dept_name, building, budget)
- instructor(ID, name, salary)
- course(course_id, title, credits)
- section(course_id, sec_id, semester, year)
- student(ID, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- inst_dept(ID, dept_name)
- stud_dept(ID, dept_name)
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- course_dept(course_id, dept_name)
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- sec_course(course_id, sec_id, semester, year)
- prereq(course_id, prereq_id)
- sec_class(course_id, sec_id, semester, year, building, room_num)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



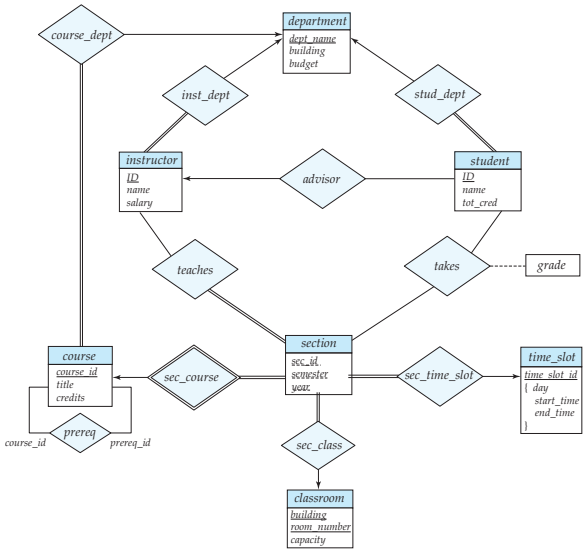
- department(dept_name, building, budget)
- instructor(ID, name, salary)
- course(course_id, title, credits)
- section(course_id, sec_id, semester, year)
- student(ID, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- inst_dept(ID, dept_name)
- stud_dept(ID, dept_name)
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- course_dept(course_id, dept_name)
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- ~~sec_course(course_id, sec_id, semester, year)~~
- prereq(course_id, prereq_id)
- sec_class(course_id, sec_id, semester, year, building, room_num)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



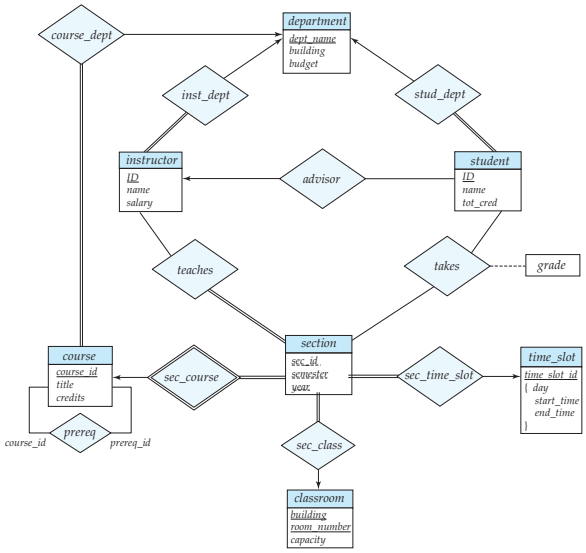
- department(dept_name, building, budget)
- instructor(ID, dept_name, name, salary)
- course(course_id, title, credits)
- section(course_id, sec_id, semester, year)
- student(ID, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- ~~inst_dept(ID, dept_name)~~
- stud_dept(ID, dept_name)
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- course_dept(course_id, dept_name)
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- ~~sec_course(course_id, sec_id, semester, year)~~
- prereq(course_id, prereq_id)
- sec_class(course_id, sec_id, semester, year, building, room_num)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



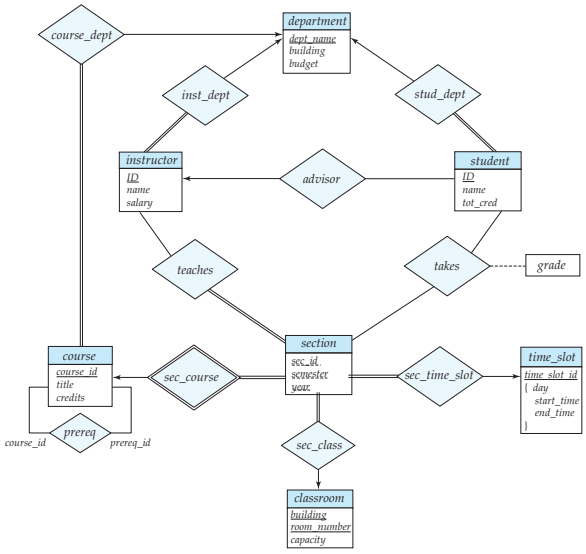
- department(dept_name, building, budget)
- instructor(ID, dept_name, name, salary)
- course(course_id, title, credits)
- section(course_id, sec_id, semester, year)
- student(ID, dept_name, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- ~~inst_dept(ID, dept_name)~~
- ~~stud_dept(ID, dept_name)~~
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- course_dept(course_id, dept_name)
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- ~~sec_course(course_id, sec_id, semester, year)~~
- prereq(course_id, prereq_id)
- sec_class(course_id, sec_id, semester, year, building, room_num)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



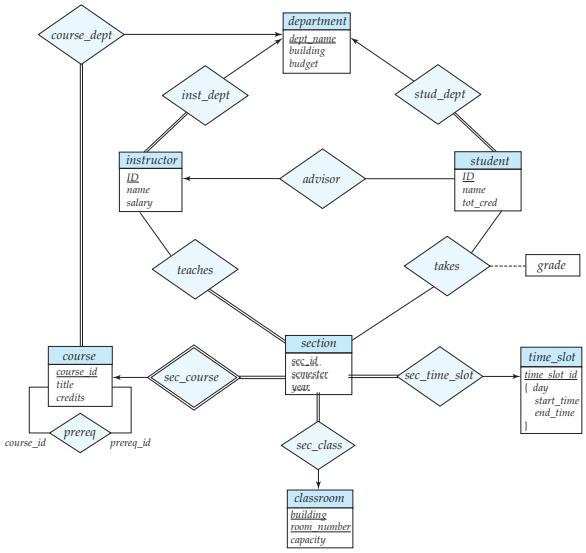
- department(dept_name, building, budget)
- instructor(ID, dept_name, name, salary)
- course(course_id, title, credits, dept_name)
- section(course_id, sec_id, semester, year)
- student(ID, dept_name, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- ~~inst_dept(ID, dept_name)~~
- ~~stud_dept(ID, dept_name)~~
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- ~~course_dept(course_id, dept_name)~~
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- ~~sec_course(course_id, sec_id, semester, year)~~
- prereq(course_id, prereq_id)
- sec_class(course_id, sec_id, semester, year, building, room_num)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



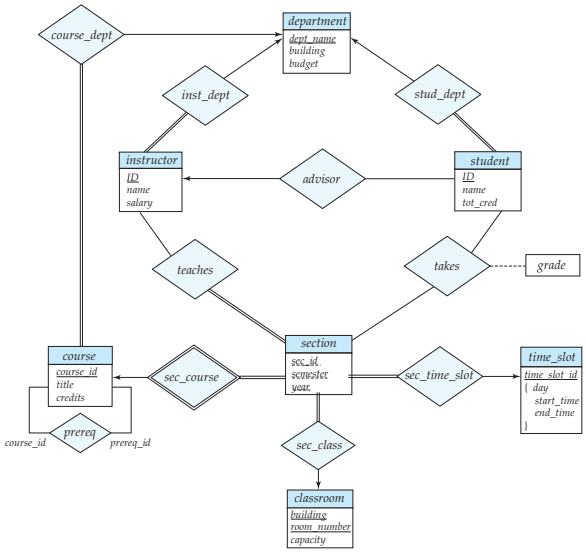
- department(dept_name, building, budget)
- instructor(ID, dept_name, name, salary)
- course(course_id, title, credits, dept_name)
- section(course_id, sec_id, semester, year, building, room_num)
- student(ID, dept_name, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- ~~inst_dept(ID, dept_name)~~
- ~~stud_dept(ID, dept_name)~~
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- ~~course_dept(course_id, dept_name)~~
- sec_time_slot(course_id, sec_id, semester, year, slot_id)
- ~~sec_course(course_id, sec_id, semester, year)~~
- prereq(course_id, prereq_id)
- ~~sec_class(course_id, sec_id, semester, year, building, room_num)~~

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



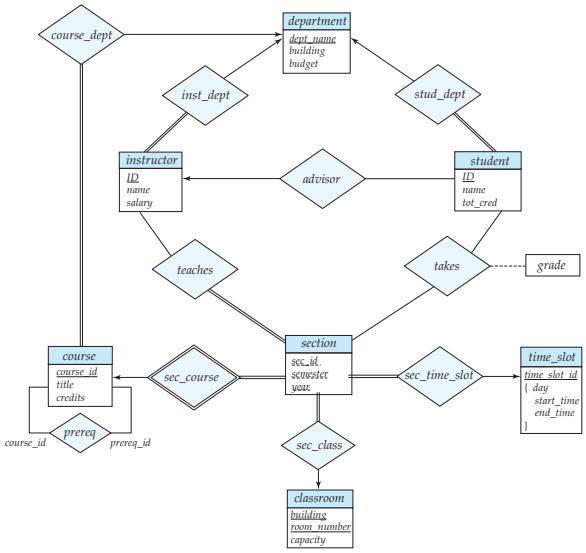
- department(dept_name, building, budget)
- instructor(ID, dept_name, name, salary)
- course(course_id, title, credits, dept_name)
- section(sec_id, course_id, semester, year, building, room_num, slot_id)
- student(ID, dept_name, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- ~~inst_dept(ID, dept_name)~~
- ~~stud_dept(ID, dept_name)~~
- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- ~~course_dept(course_id, dept_name)~~
- ~~sec_time_slot(course_id, sec_id, semester, year, slot_id)~~
- ~~sec_course(course_id, sec_id, semester, year)~~
- prereq(course_id, prereq_id)
- ~~sec_class(building, room_num, capacity, building, room_num)~~

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set



- department(dept_name, building, budget)
- instructor(ID, dept_name, name, salary)
- course(course_id, title, credits, dept_name)
- section(sec_id, course_id, semester, year, building, room_num, slot_id)
- student(ID, dept_name, name, tot_cred)
- classroom(building, room_num, capacity)
- time_slot(slot_id, day, start_time, end_time)

- teaches(ID, course_id, sec_id, semester, year)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(i_id, s_id)
- prereq(course_id, prereq_id)

- lots of foreign key dependences (weak, relationships..)
- also, we only allow one time slot

Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g., A ternary relationship ***parents***, relating a child to his/her father and mother, is best replaced by two binary relationships, ***parent1*** and ***parent2***
    - Using two binary relationships allows partial information (e.g., only parent1 being known)
  - But there are some relationships that are naturally non-binary
    - Example: ***proj_group***, with several project members

Design Issues

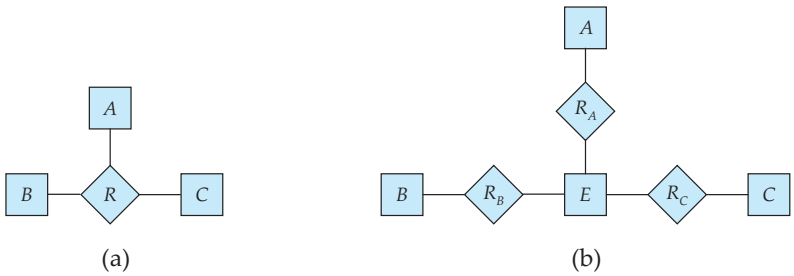
- **Binary versus n-ary relationship sets**
  Although it is possible to replace any nonbinary (*n*-ary, for *n* > 2) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.

- **Placement of relationship attributes can be tricky**

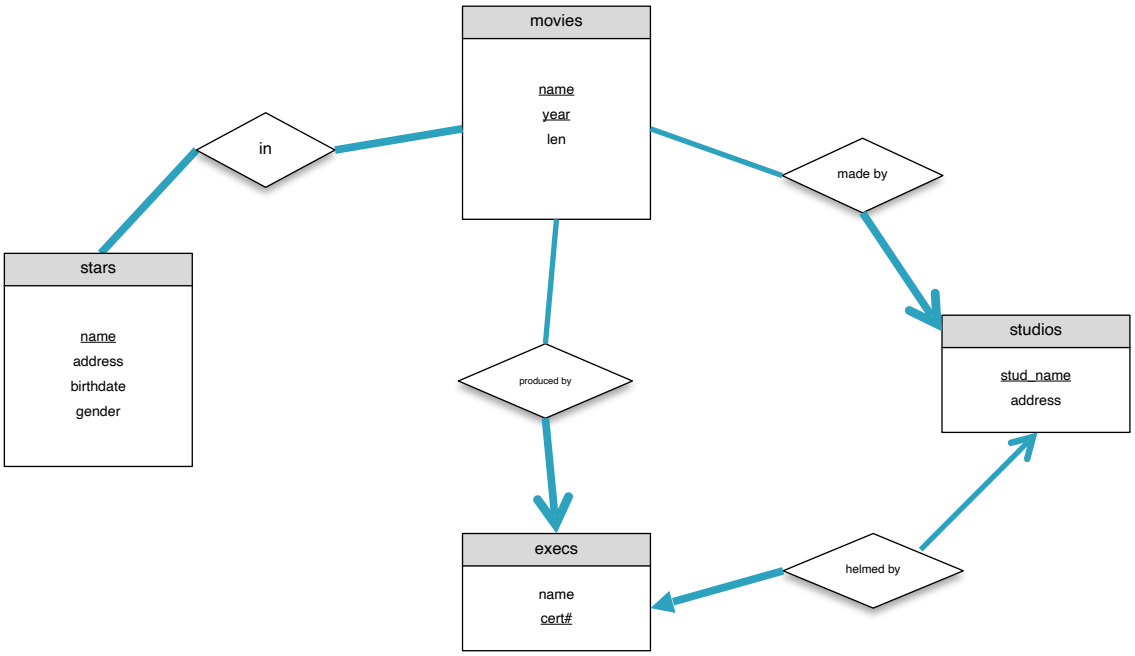
  e.g., attribute ***date*** as attribute of ***advisor*** or as attribute of ***student***

Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace R between entity sets A, B and C by an entity set E, and R_A, R_B, R_C, relating E with A, B, and C
  - Create a special identifying attribute for E
  - Add any attributes of R to E
  - For each relationship (a_i, b_i, c_i) in R
    - create a new entity e_i in the entity set E
    - add (e_i, a_i) to R_A, etc.



A Movie Industry Schema

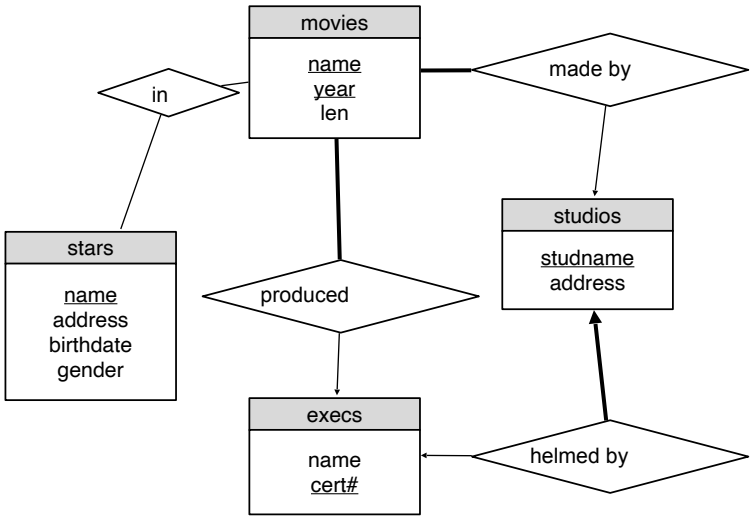


x

Outline

- Relational Algebra (6.1)
- E/R Model (7.2 - 7.4)
- E/R Diagrams (7.5)
- Reduction to Schema (7.6)
- Relational Database Design (7.7)
- Functional Dependencies (8.1 – 8.4)
- Normalization (8.5 – 8.7)

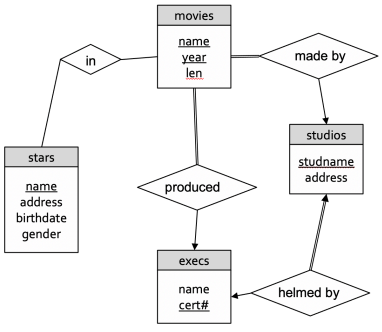
A Movie Industry Schema



Relational Schemas and Redundancy

- movies(name, year, len)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address)

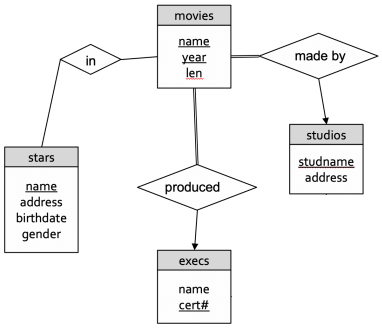
- in(star_name, movie_name, movie_year)
- made_by(movie_name, movie_year, studioname)
- produced_by(movie_name, movie_year, cert#)
- helmed_by(cert#, stud_name)



Relational Schemas and Redundancy

- movies(name, year, len)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, **pres#**)

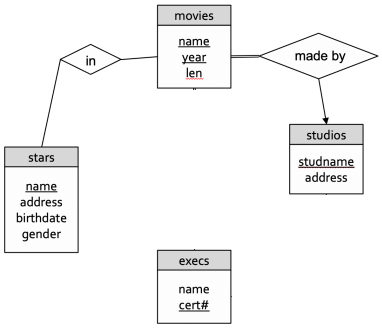
- in(star_name, movie_name, movie_year)
- made_by(movie_name, movie_year, studioname)
- produced_by(movie_name, movie_year, cert#)



Relational Schemas and Redundancy

- movies(name, year, len, **prod#**)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

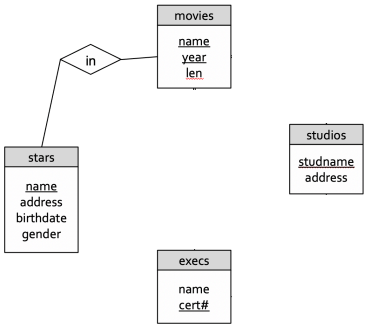
- in(star_name, movie_name, movie_year)
- made_by(movie_name, movie_year, studioname)



Relational Schemas and Redundancy

- movies(name, year, len, prod#, **studio_name**)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

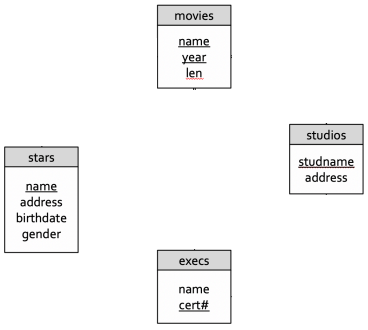
- in(star_name, movie_name, movie_year)



Relational Schemas and Redundancy

- movies(name, year, len, prod#, studio_name, **star_name**)
- stars(name, addr, gender, birthdate)
- execs(name, cert#)
- studios(stud_name, address, pres#)

Is this a good idea???



Relational Database Design

or

"Troubles With Schemas"

Movie(*title, year*, length, inColor, studioName, producerC#, starName)

Title	Year	Length	inColor	StudioName	prodC#	StarName
Star wars	1977	121	Yes	Fox	128	Hamill
Star wars	1977	121	Yes	Fox	128	Fisher
Star wars	1977	121	Yes	Fox	128	H. Ford
King Kong	2005	187	Yes	Universal	150	Watts
King Kong	1933	100	no	RKO	20	Fay

Issues:

1. Redundancy ➔ higher storage,
2. Inconsistencies ("anomalies")
   update anomalies, insertion anomalies
3. Need nulls!
   movies w/o actors, pre-productions, etc

Movie(*title, year*, length, inColor, studioName, producerC#, starName)

Title	Year	Length	inColor	StudioName	prodC#	StarNames
Star wars	1977	121	Yes	Fox	128	{Hamill, Fisher, H. Ford}
King Kong	2005	187	Yes	Universal	150	Watts
King Kong	1933	100	no	RKO	20	Fay

Issues:

3. Avoid sets
   - Hard to represent
   - Hard to query

Less Redundancy through Decomposition

Split Studio(*name*, address, presC#) into:

Studio1 (name, presC#),

Studio2(name, address)???

Name	presC#
Fox	101
Studio2	101
Universial	102

Name	Address
Fox	Address1
Studio2	Address1
Universial	Address2

This process is also called *"decomposition"*

Issues:

4. Requires more joins (w/o any obvious benefits)
5. Hard to check for some dependencies

   What if the "address" is actually the presC#'s address ?

   No easy way to ensure that constraint (w/o a join).

Are smaller schemas always good ????

Decompose StarsIn(movieTitle, movieYear, starName) into:

StarsIn1(movieTitle, movieYear)

StarsIn2(movieTitle, starName) ???

movieTitle	movieYear
Star wars	1977
King Kong	1933
King Kong	2005

movieTitle	starName
Star Wars	Hamill
King Kong	Watts
King Kong	Faye

Issues:

6. "joining" them back results in more tuples than what we started with

(King Kong, 1933, Watts) & (King Kong, 2005, Faye)

This is a "lossy" decomposition

We lost some constraints/information

The previous example was a "lossless" decomposition.

Desiderata

- No sets
- Correct and faithful to the original design
  - Avoid lossy decompositions
- As little redundancy as possible
  - To avoid potential anomalies
- No "inability to represent information"
  - Nulls shouldn't be required to store information
- Dependency preservation
  - Should be possible to check for constraints

Not always possible.

We sometimes relax these for:

simpler schemas, and *fewer joins during queries.*

Relational Database Design

- Where did we come up with the schema that we used ?
  - E.g. why not store the actor names with movies ?
- If from an E-R diagram, then:
  - Did we make the right decisions with the E-R diagram ?
- Goals:
  - Formal definition of what it means to be a “good” schema.
  - How to achieve it.

Outline

- Mechanisms and definitions to work with FDs
  - Closures, candidate keys, canonical covers etc…
  - Armstrong axioms
- Decompositions
  - Loss-less decompositions, Dependency-preserving decompositions
- BCNF
  - How to achieve a BCNF schema
- BCNF may not preserve dependencies
- 3NF: Solves the above problem
- BCNF allows for redundancy
- 4NF: Solves the above problem

Approach

- 1. We will encode and list all our knowledge about the schema
  - Functional dependencies (FDs)

    SSN → name (means: SSN "determines" name)

    If two tuples have the same "SSN", they must have the same "name"
  - But:

    movietitle → length (Not true)

    (movietitle, movieYear, movieDirector) → length (True)
- 2. We will define a set of rules that the schema must follow to be "good"
  - "Normal forms": 1NF, 2NF, 3NF, BCNF, 4NF, …
  - A normal form specifies constraints on the schemas and FDs
- 3. If not in a "normal form", we modify the schema

FDs: Example

Course ID	Course Name	Dept Name	Credits	Section ID	Semester	Year	Building	Room No.	Capacity	Time Slot ID

Functional dependencies:

- course_id →
- building, room_num →
- course_id, section_id, semester, year →

Functional Dependencies

- Let r(*R*) be a relation schema and

  $\alpha \subseteq R$ *and* $\beta \subseteq R$

- The *functional dependency*

  $\alpha \rightarrow \beta$

  holds on *R* iff for any *legal* relations *r(R)*, whenever two tuples t_1 and t_2 of *r* have same values for α, they have same values for β.

  $t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$

- Example:

With *r(R)*:
- *r* is a relation (w/ tuples)
- *R* is *r*'s "schema", i.e. set of attributes

An *instance* is the value of a *r* at a particular point in time.

- On this instance, *A* → *B* does **NOT** hold, but *B* → *A* does hold.

A	B
1	4
1	5
3	7