Lecture 2: Database Design / Data Modeling

Definition: *Database design* is the activity of specifying the schema of a database in a given data model.

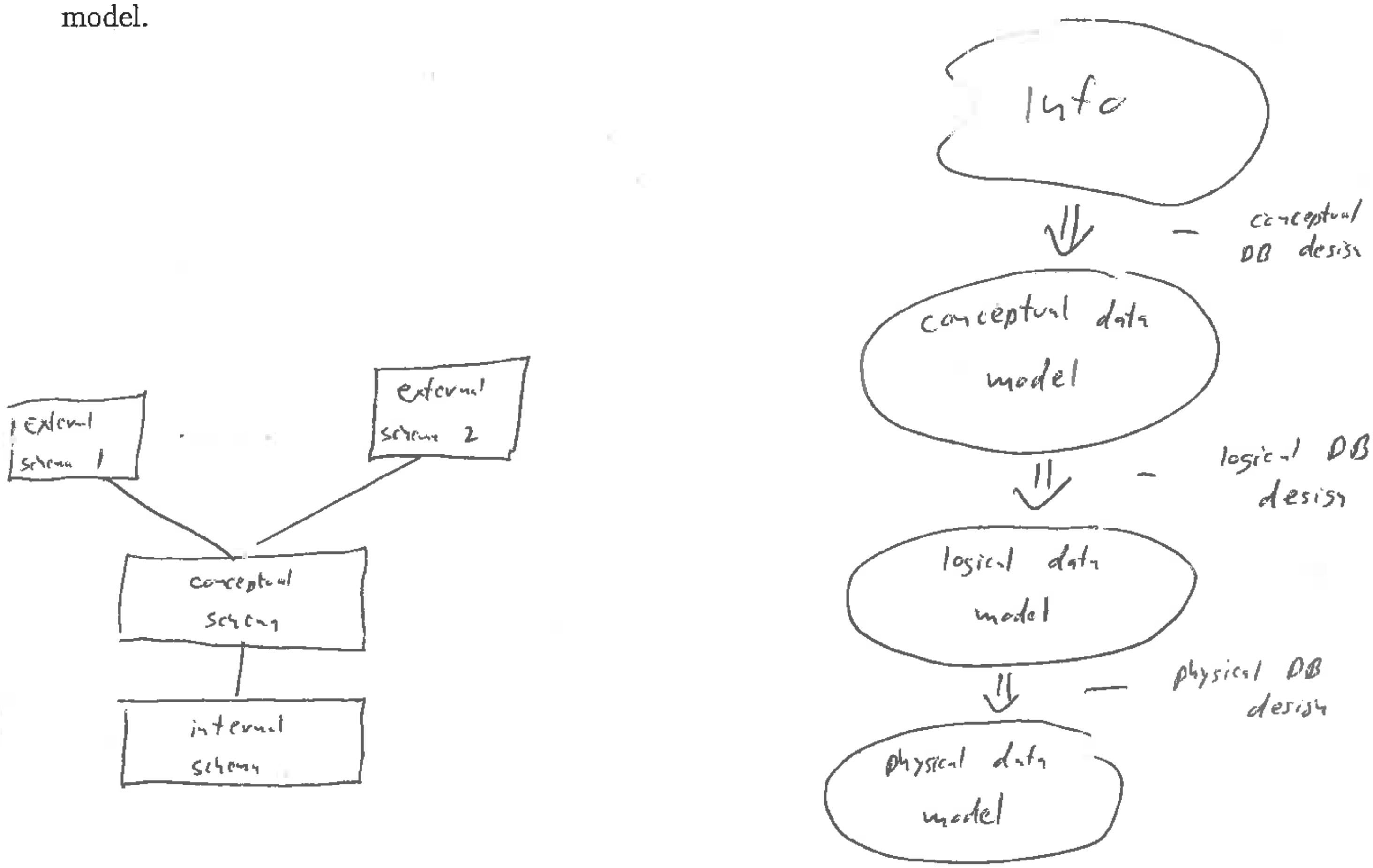


Figure 1: Process of Database Design

Conceptual Database Design:

- Thinking about the data.
- Identifying semantic objects and relationships among them.
- *Goal:* Abstract but complete description of the database in a pseudo-formal notation. (paper and pencil design)

Logical Database Design:

- Transforming the initial conceptual design into a formal schema expressed in the chosen data model, i.e. deriving the ~~conceptual~~ logical schema.

Physical Database Design:

- Specifying access methods, i.e. the internal storage structures of the objects implementing the ~~conceptual~~ logical schema, i.e. deriving the internal schema.

WHY DATABASE DESIGN?

Conceptual Database Design

- Semantic clarity.

Logical Database Design

- Schema together with DML determines the application program.
- Some schemas (or a powerful DML) make queries simpler.
- With the appropriate schema one avoids redundancy, update anomalies, etc.

Physical Database Design

- Performance
- Support query optimizer.

Conceptual Database Design

The consensus choice for the data model used is the *Entity-Relationship* (E-R) model.

Two semantic primitives:

- *Entities:* Things that exist and are distinguishable, e.g. Paul, Sue, Sue's car, Carnegie Hall, etc.
- *Relationships:* Two or more entities may participate in a relationship, e.g. Paul *borrowed* Sue's car, Sue *is a patron of* Carnegie Hall.

Entities and relationships may have *attributes*, e.g. Paul has an age, Sue's car has a model, Sue is a $10000 patron of Carnegie Hall.

A group of all similar (same properties) entities form an *entity set.* E.g. All persons, all cars, all concert halls.

Relationships can also be grouped into sets that are homogeneous in participating entity sets. Such a group is called a *relationship set.* E.g. All (person, car) pairs such that the person owns the car.

The overall structure of a database can be expressed graphically by an E-R diagram, which consists of the following components:

- Rectangles for entity sets.
- Diamonds for relationship sets.
- Ellipses for attributes.
- Links for associating attributes to entity and relationship sets, and entity sets to relationship sets.

Example: A library database has authous that have written books about various subjects (one author per book). It also has info about libraries that carry books on various subjects.

Have to identify entity and relationship sets. 1st attempt:

What is wrong? (When designing model must keep in mind what kind of questions will be asked.)

- How to represent the fact that a library carries books of a specific author.
- Where to put info about which edition of a book is carried by a library, how many copies, etc.?

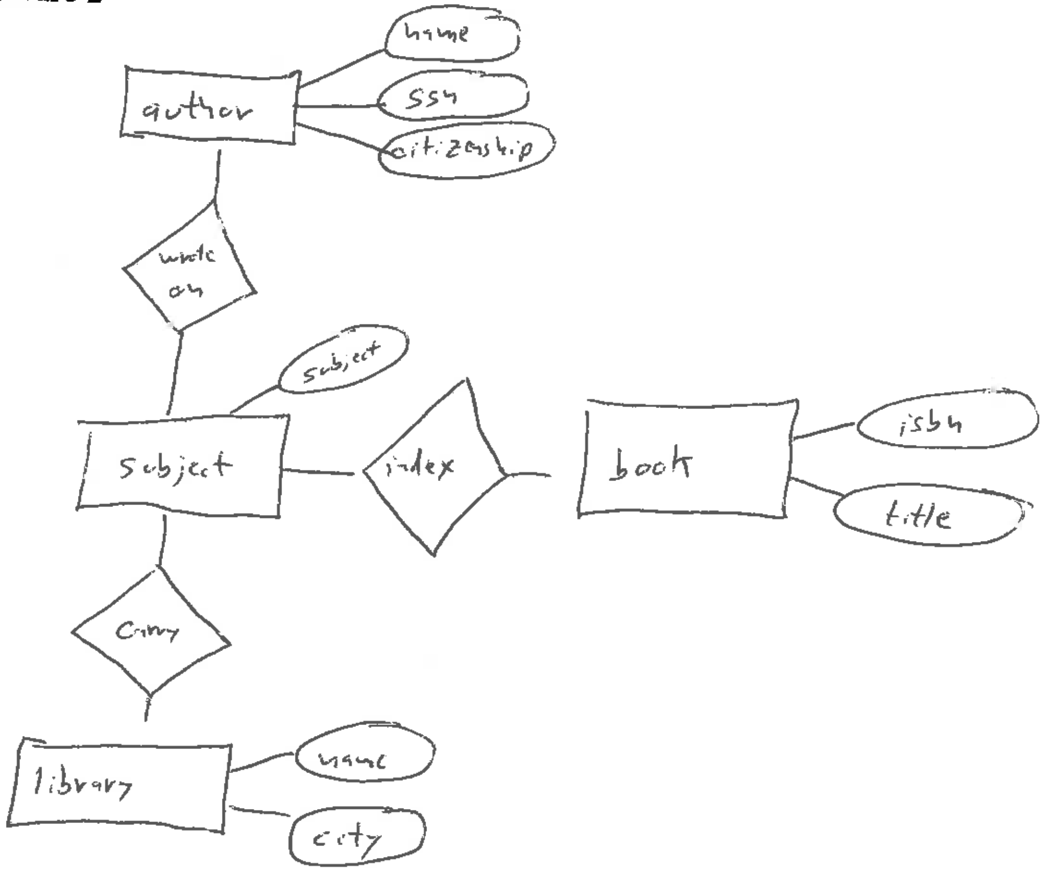


Figure 2: 1st attempt E-R model

- ~~Can we find out whether a library carries books by a certain author?~~

Which books has "Leitenesscu" written

2nd attempt:

To find whether a library carries books by an author just start with the author, get his/her books and see which library they are in.

The "carry" relationship set also has attributes: how may copies, which edition.

Must also include constraint of only one author per book: ⇒ figure 4

A relationship is an *ordered* set of entities. So, a reslationship set is a set of these ordered sets. E.g. (Paul, Sue's car) is a member of the relationship set **borrowed** = { (Paul, Sue's car), (Mary, Alice's saw),(Tom, Jerry's cat)}.

A relationship may involve:

- *2 entities*: binary relationship
- *3 entities*: ternary relationship
- *N entities*: N-ary relationship

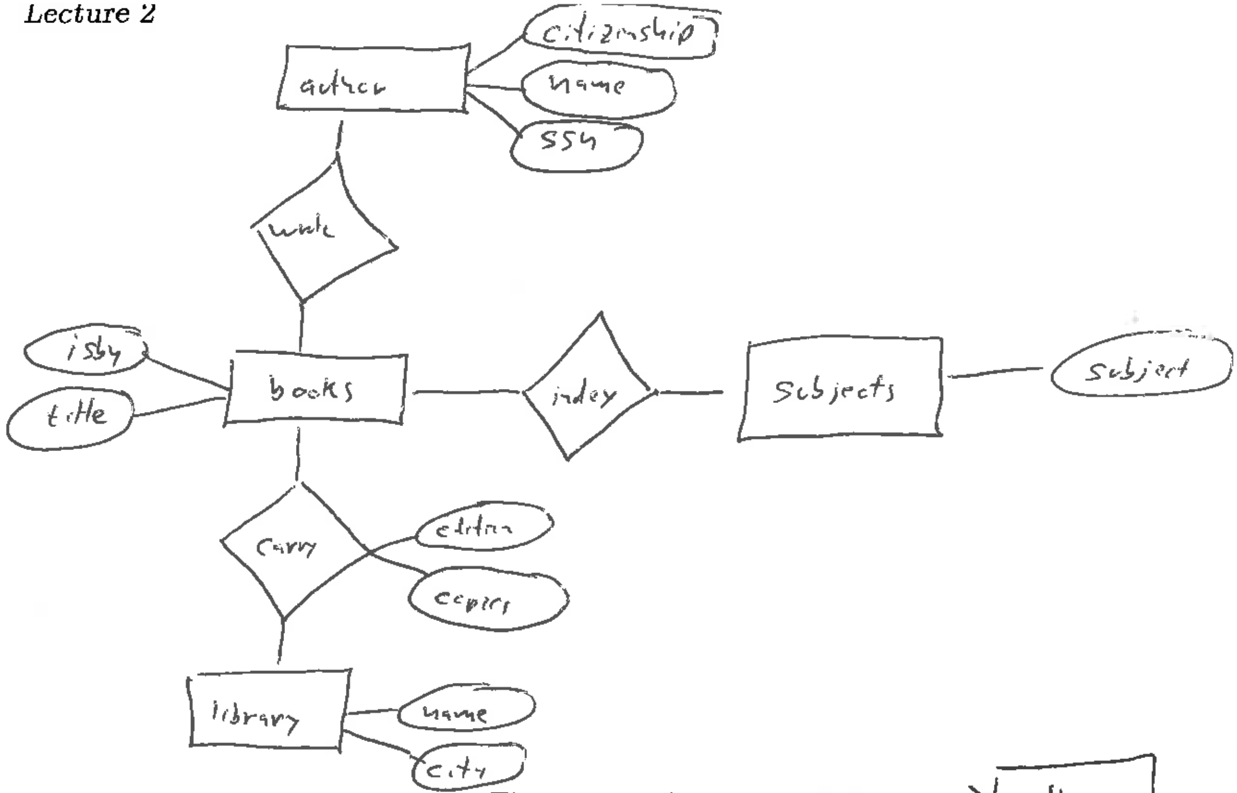


Figure 3: 2nd attempt E-R model

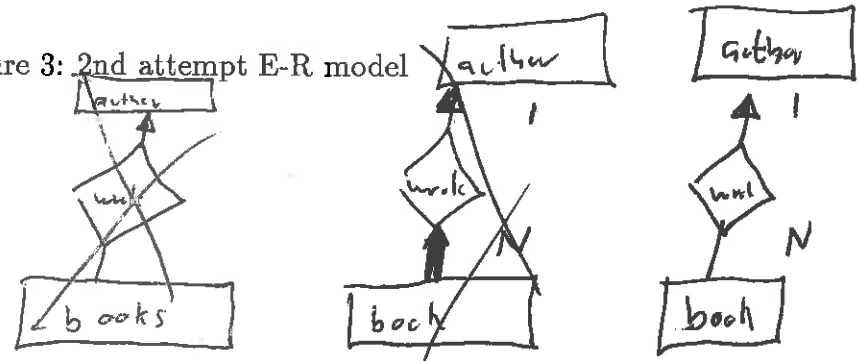
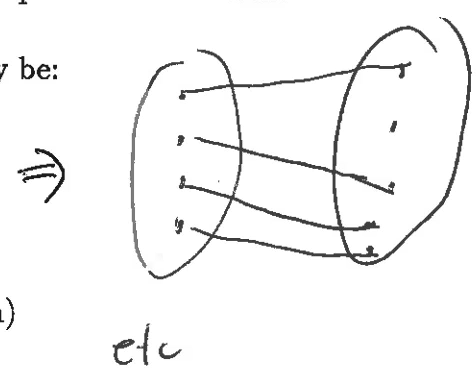


Figure 4: Single author per book constraint

A binary relationship set between sets A and B may be:

- **1:1** Persons having social Security numbers.
- **N:1** Children having fathers. (function)
- **1:N** Fathers having children. (inverse function)
- **M:N** Students enrolling in classes.



Entities and relationships are distinguished by using various *keys*:

- **Definition**: A *superkey* is a set of one or more attributes of an entity (relationship) whose values uniquely identify an entity (relationship).

  E.g. The name "Asimov", Asimov's citizenship, and Asimov's ssn is a superkey for Faulkner as an entity.

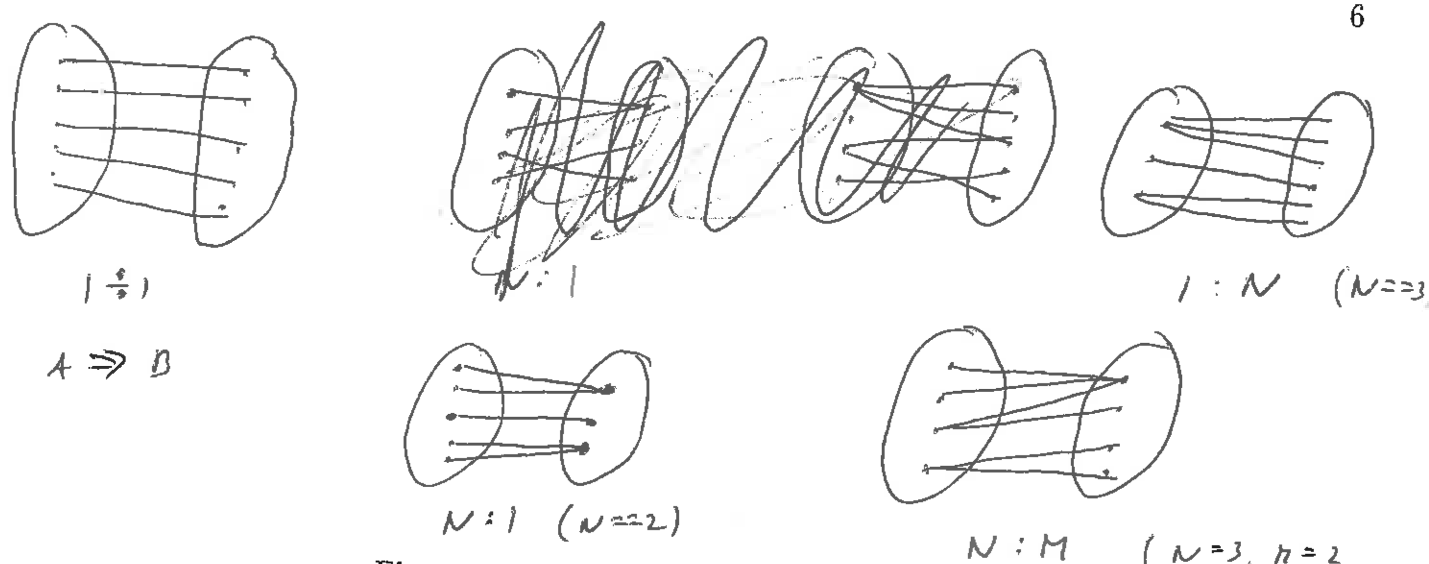


Figure 5: Binary relationship mappings

- **Definition:** A *candidate key* is a *minimal* superkey, i.e. no subset of its attributes is a superkey.
  E.g. Asimov's ssn is a candidate key.
- **Definition:** A *primary key* is one distinguished candidate key that serves as the identifier of the entity (relationship) concerned.
- **Definition:** A *foreign key* is a set of one or more attributes of an entity (relationship) that serves as a primary key of another entity or relationship.

  E.g. For the relationship of students enrolling in classes, the student ID is a foreign key.

Logical Data Models

There is no standard DML for the E-R model.

Conceptual data models are mapped into logical data models.

Virtually every logical data model represents entitiy sets as file.

File	=	Entity set
Record	=	Entity
Field	=	Attribute

The differences between data models are primarily in how they represent relationships. Two basic techniques:

- Represent relationships as files as well (relational model).
- Represent relationships as links between files (network and hierarchical models).

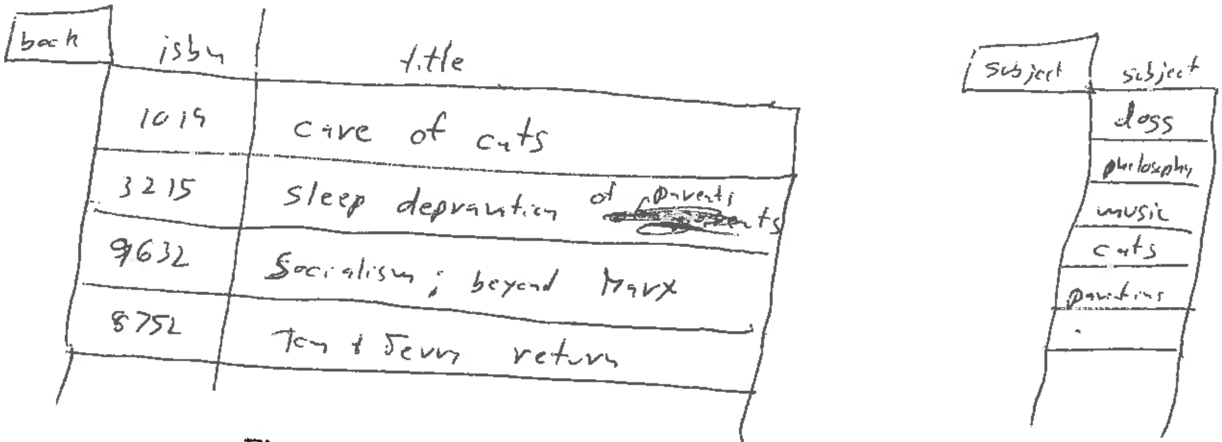


Figure 6: File representation of Entity set

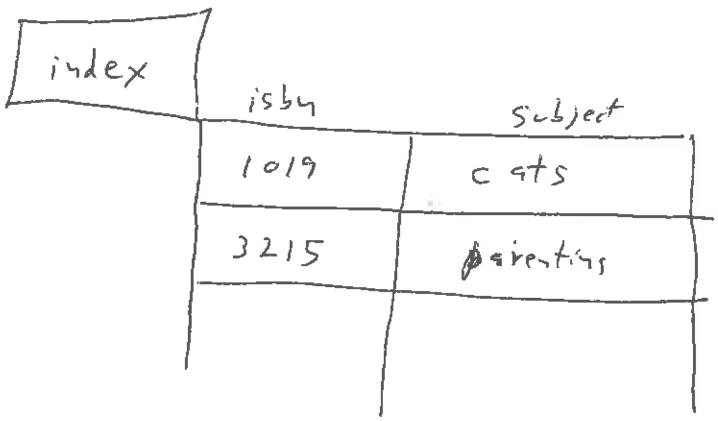


Figure 7: Relational Model Representation

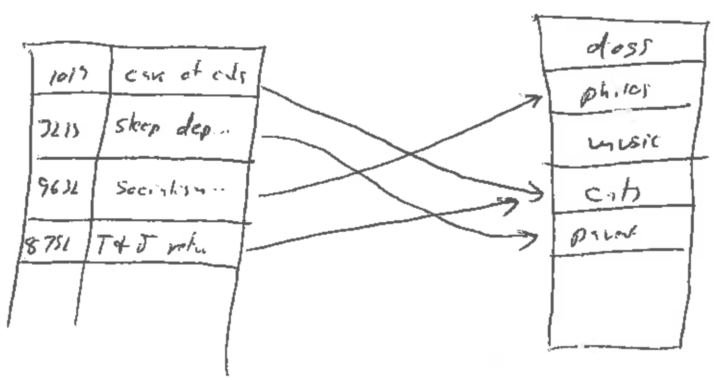


Figure 8: Network Model Representation

More E/R examples

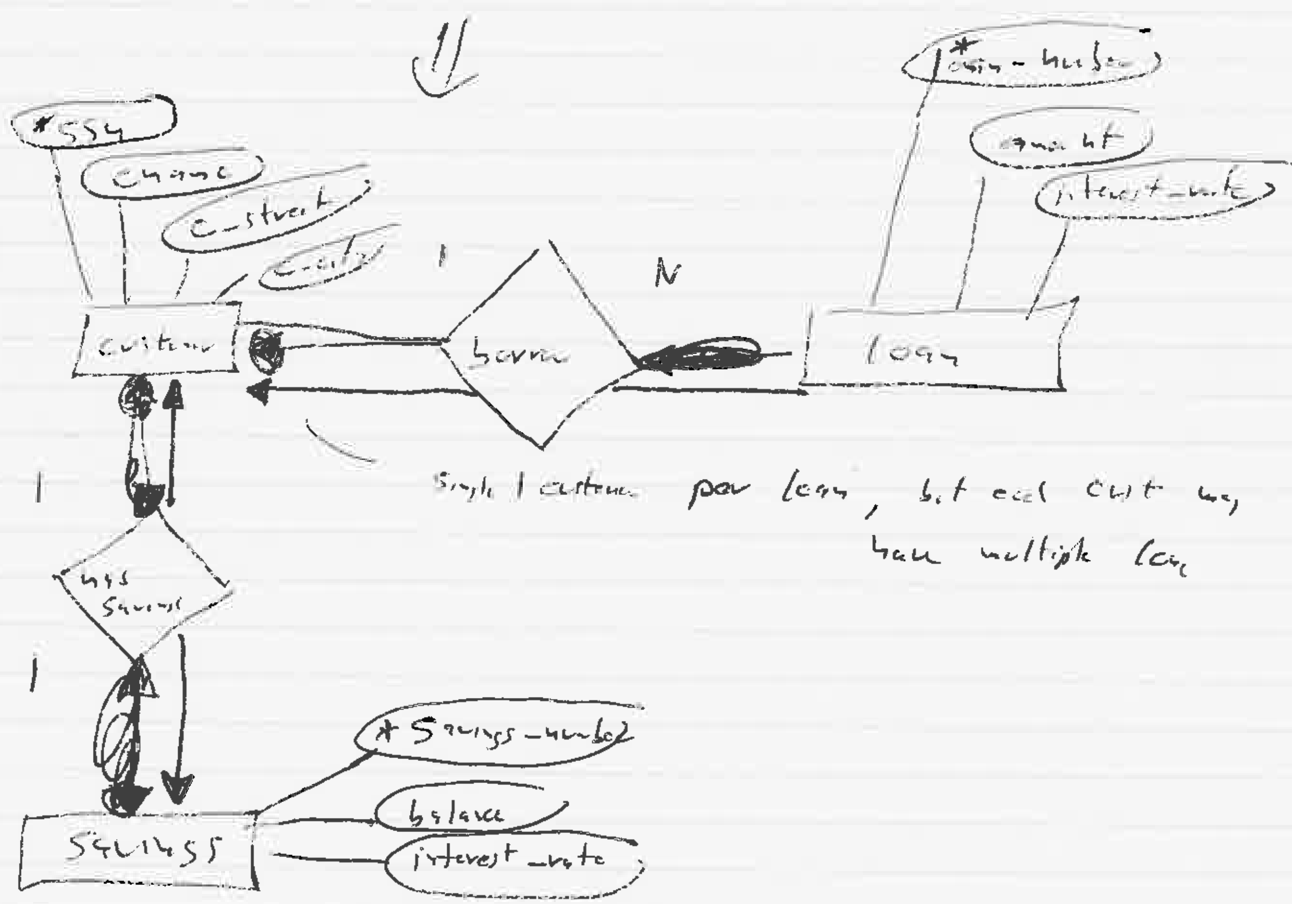
You are designing a DB for a bank.
Each customer may have savings accts, checking accts, loans.

Each customer has a (name, ssn, street, city, zip)

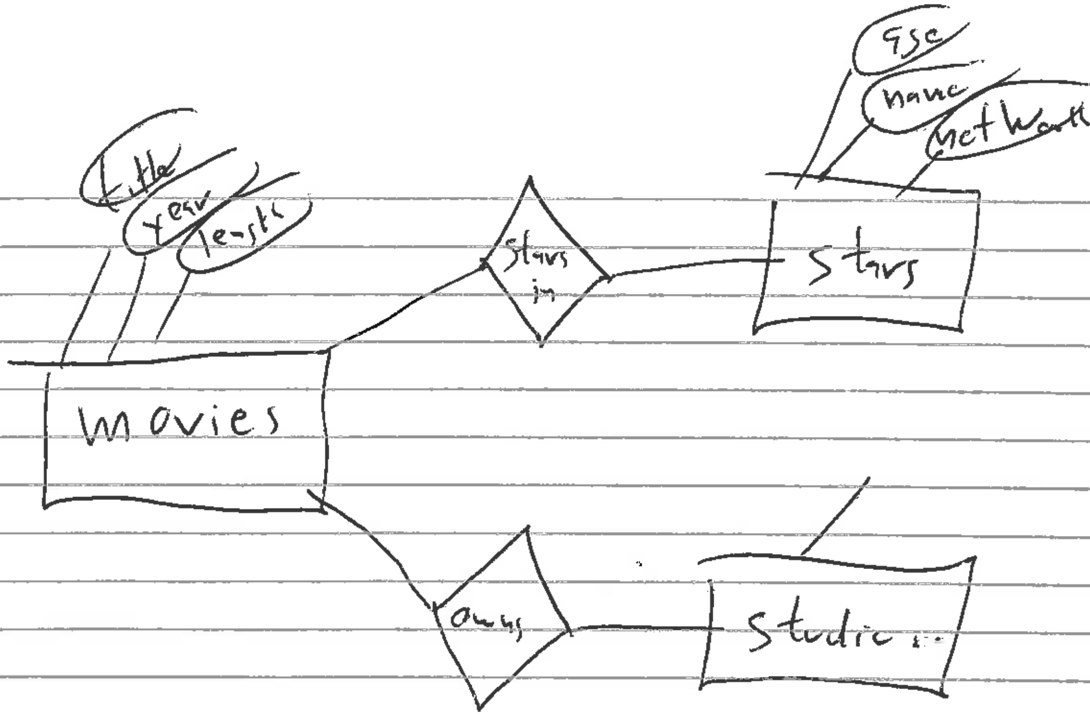
Each loan has (loan-number, amount, interest-rate)

Each saving has (savings-number, balance, interest-rate)

~~Each checking ... (... number, balance)~~



2.6 C

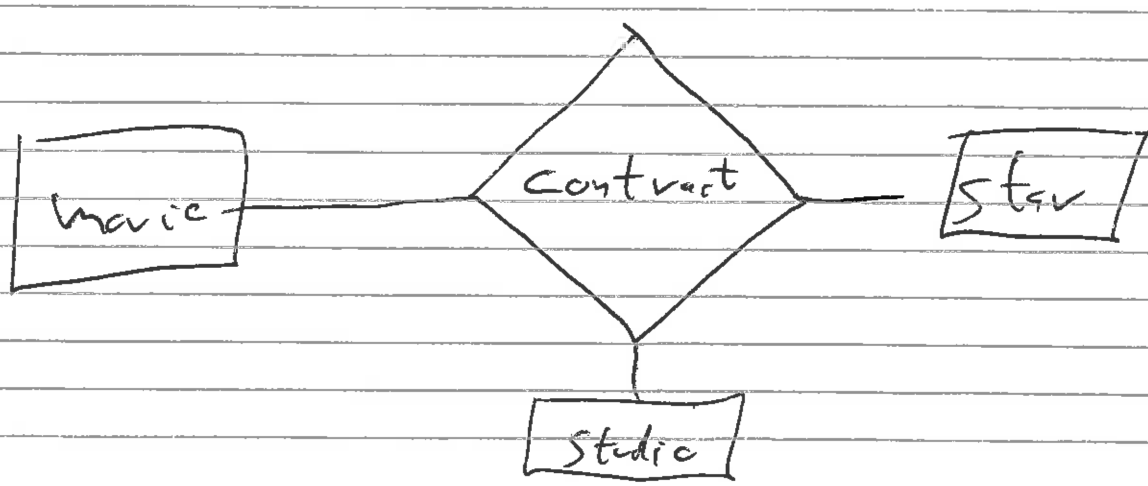


* Can a movie be owned by multiple studios?

⇒ Yes

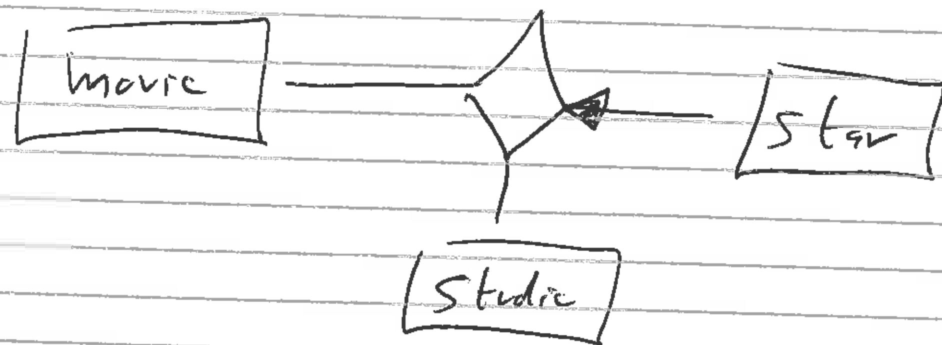
* How make so can only be owned by 1 studio?

* How can you represent a contract between star-movie-studio?



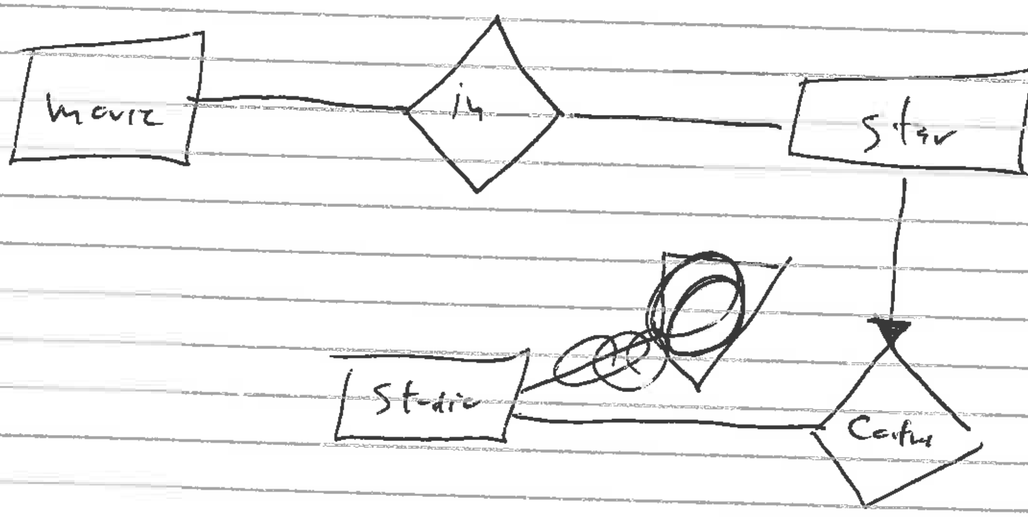
How represent a star has contracts <u>exclusively</u> with one studio?

?



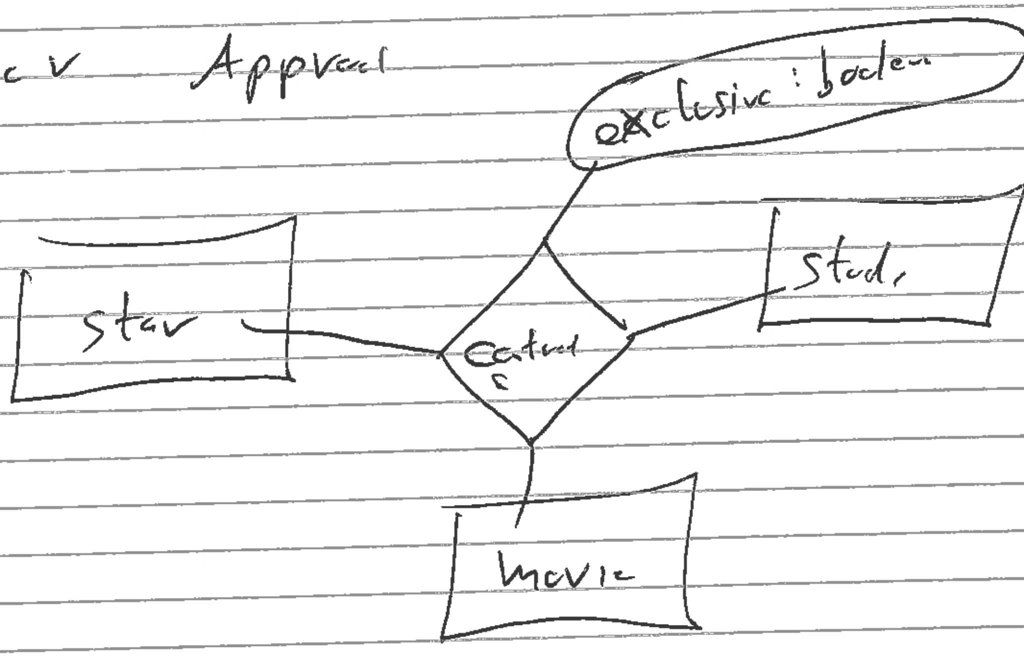
<u>No</u> this means a specific star appears only once in the relationship

Can be done more round-about



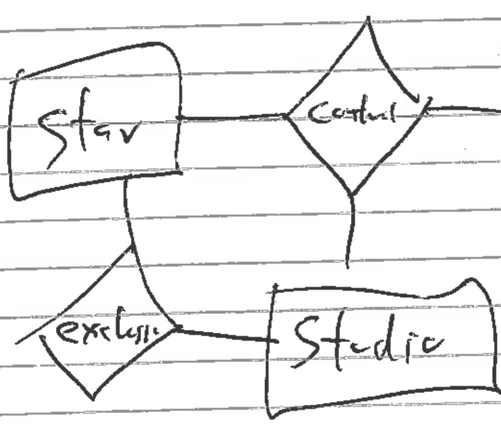
Problem: Not <u>all</u> stars must have exclusive contract

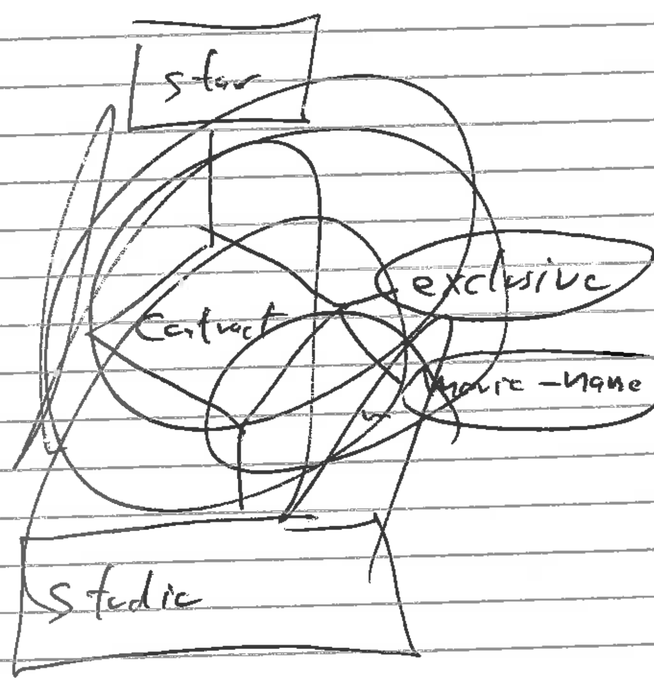
Another Approach



But this is really an attribute of star-studio

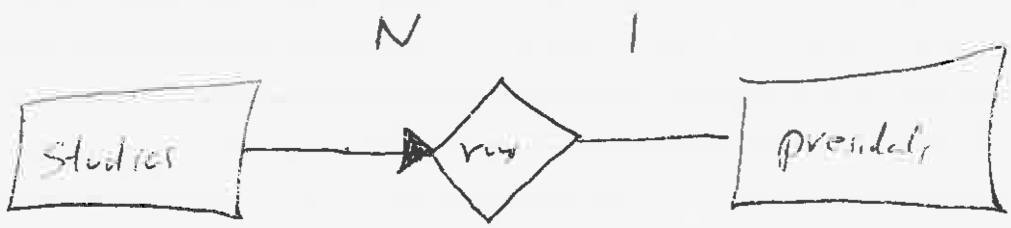
So better as





2.6f

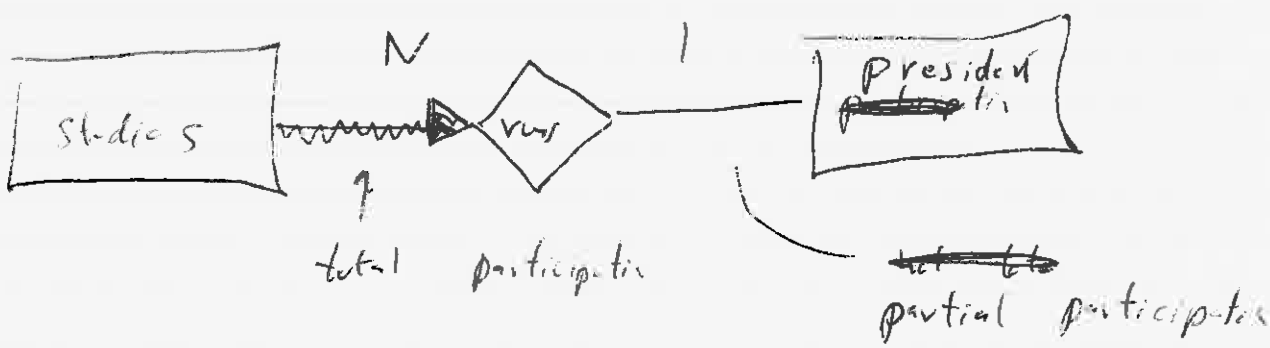
Participation constraints



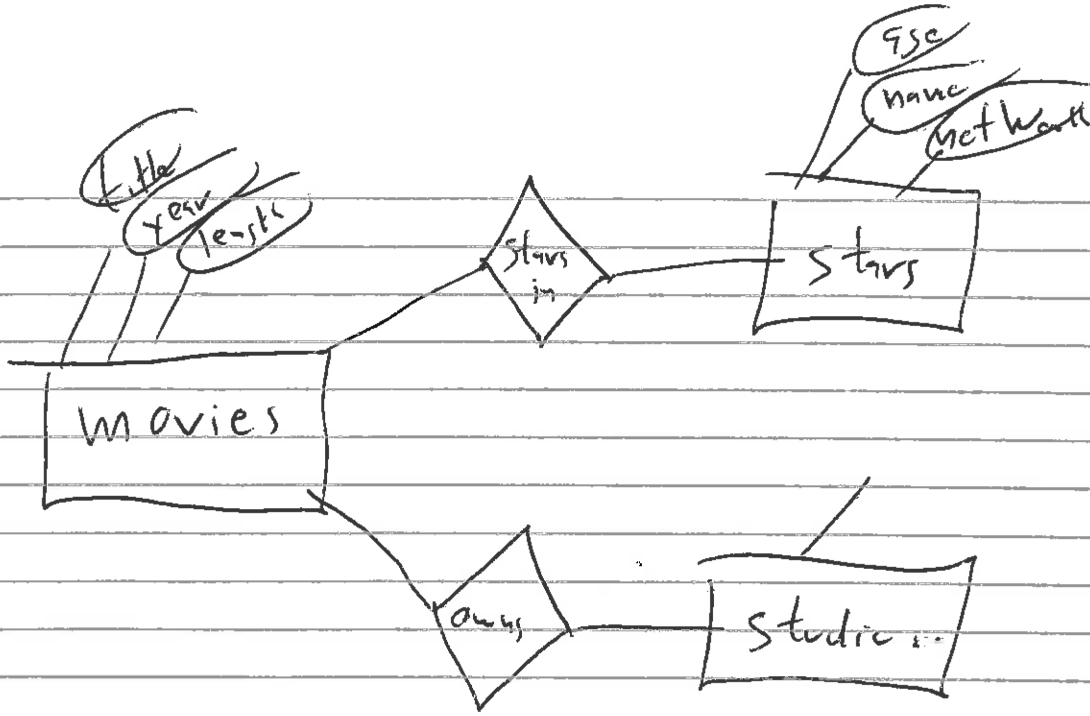
I could say every studio must have a president.

This is a <u>total participation constraint</u>

i.e. each studio entity must participate in the "runs" relationship



2.6 C

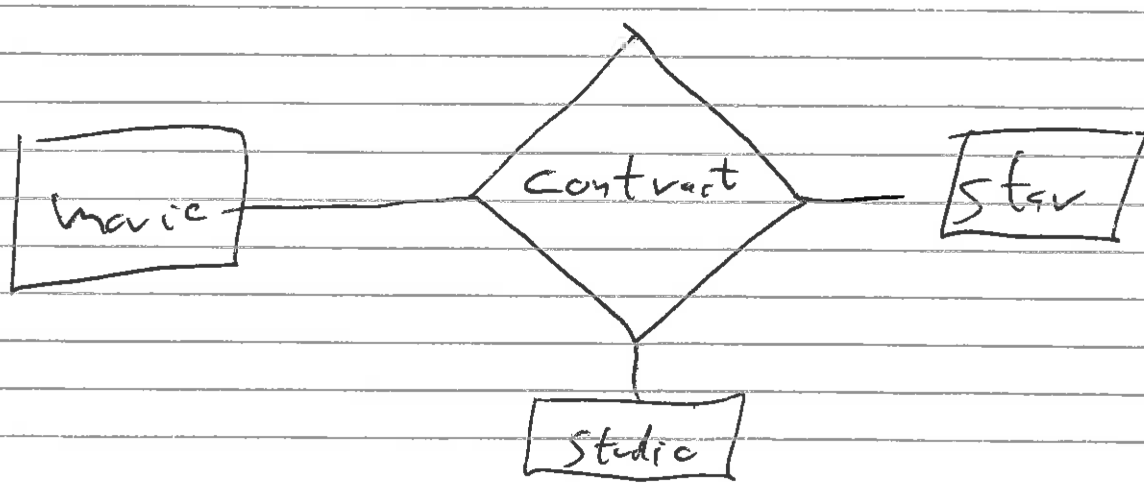


* Can a movie be owned by multiple studios?

⇒ Yes

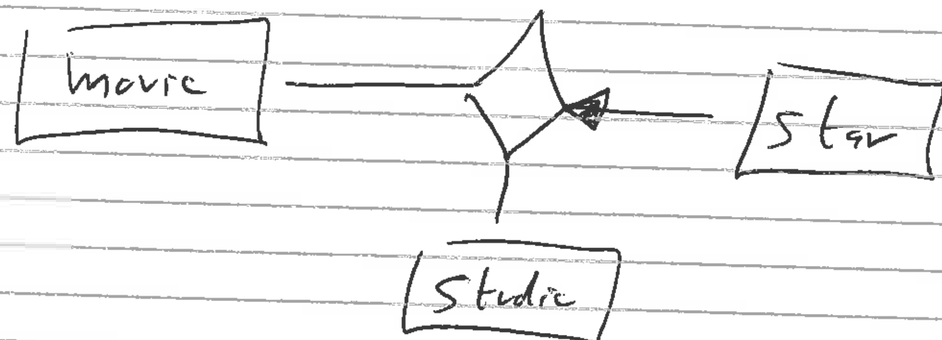
* How make so can only be owned be 1 studio?

* How can you represent a contract between star-movie-studio?



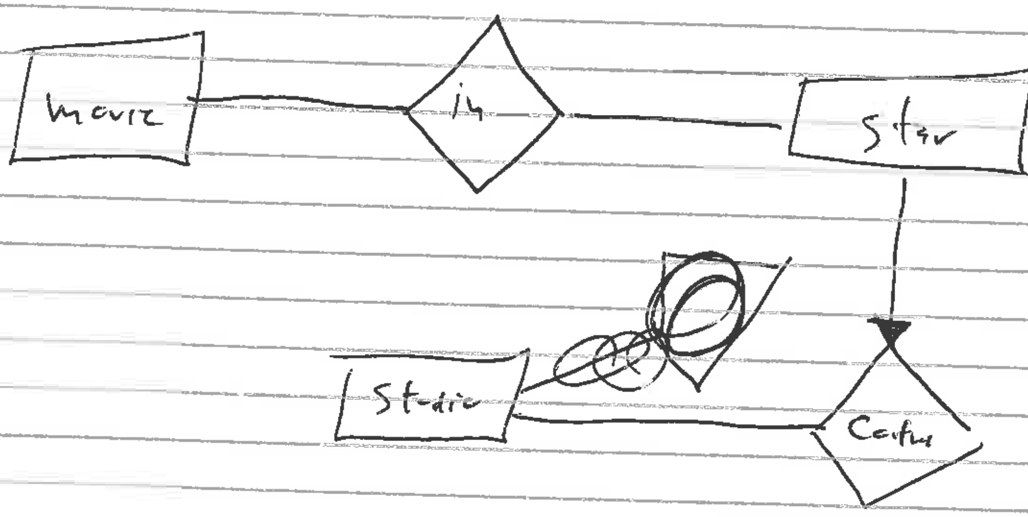
How represent a star has contracts **exclusively** with one studio?

?



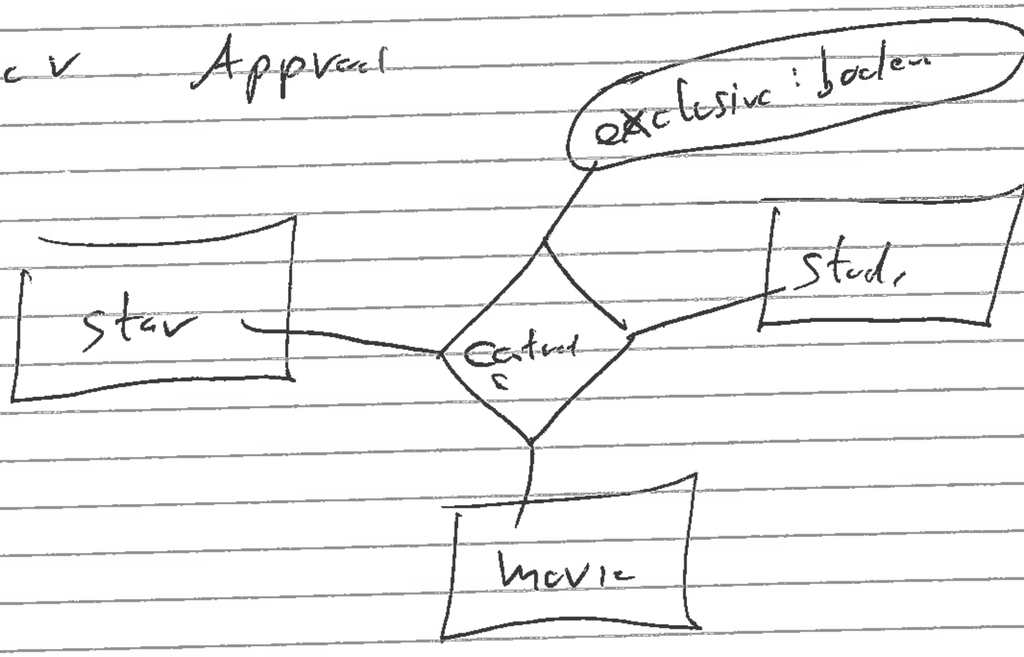
No this means a specific star appears only once in the relationship

Can be done more round-about



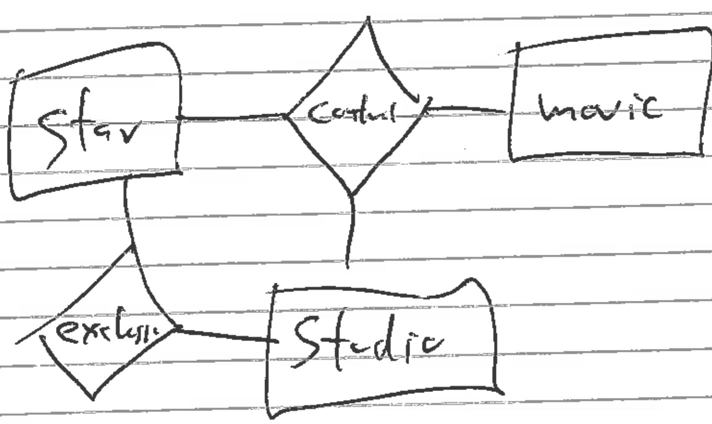
Problem: Now **all** stars must have exclusive contract

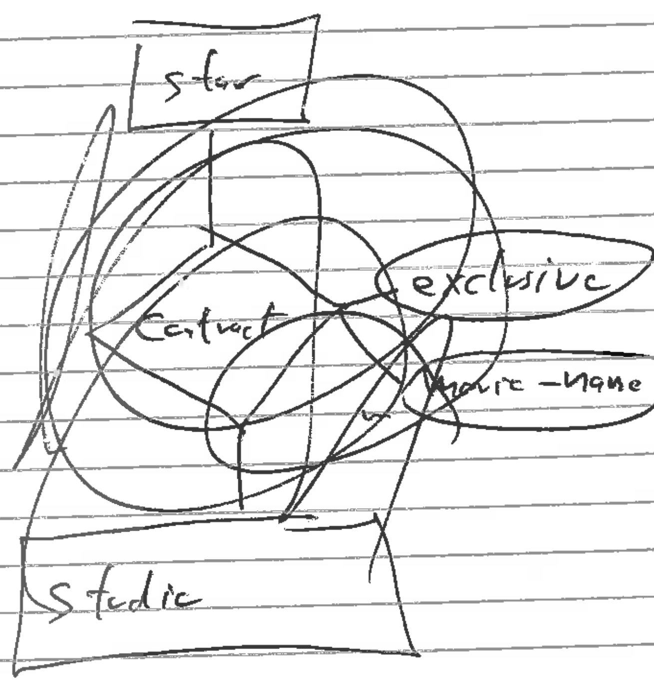
Another Approach



But this is really an attribute of star-studio

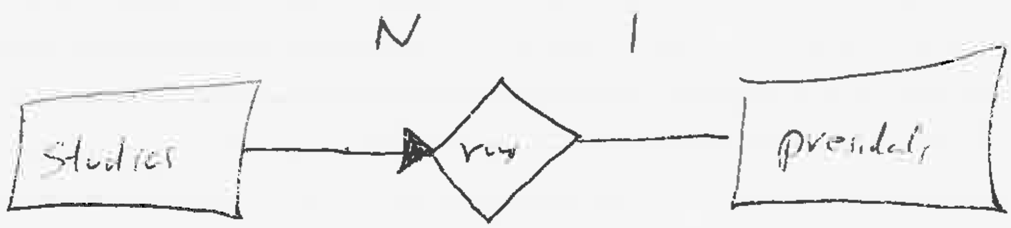
so better as





2.6f

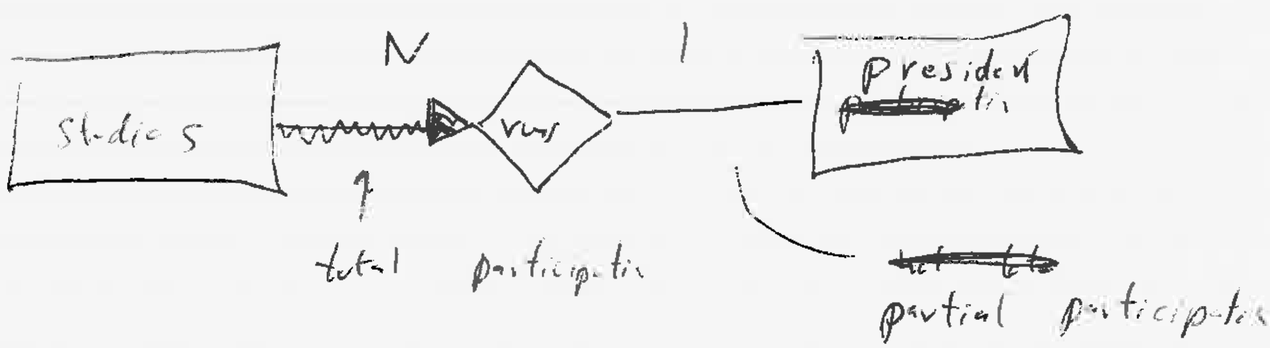
Participation constraints



I could say every studio must have a president.

This is a total participation constraint

i.e. each studio entity must participate in the "runs" relationship



Lecture 3: Relational Model

It is the most popular logical model.

Both entities and relationships are represented by *relations* (alias tables).

- Each entity is a single tuple (alias record, alias row) in the relation.
- Each attribute of the entity is a *column* (alias field) in the tuple. Recall that all entities in an entity set have the same attributes.
- Likewise, each relationship is a single tuple in the relation.
- The fields in a relationship relation are the primary keys of the participating entities plus any additional attributes the relationship might have.

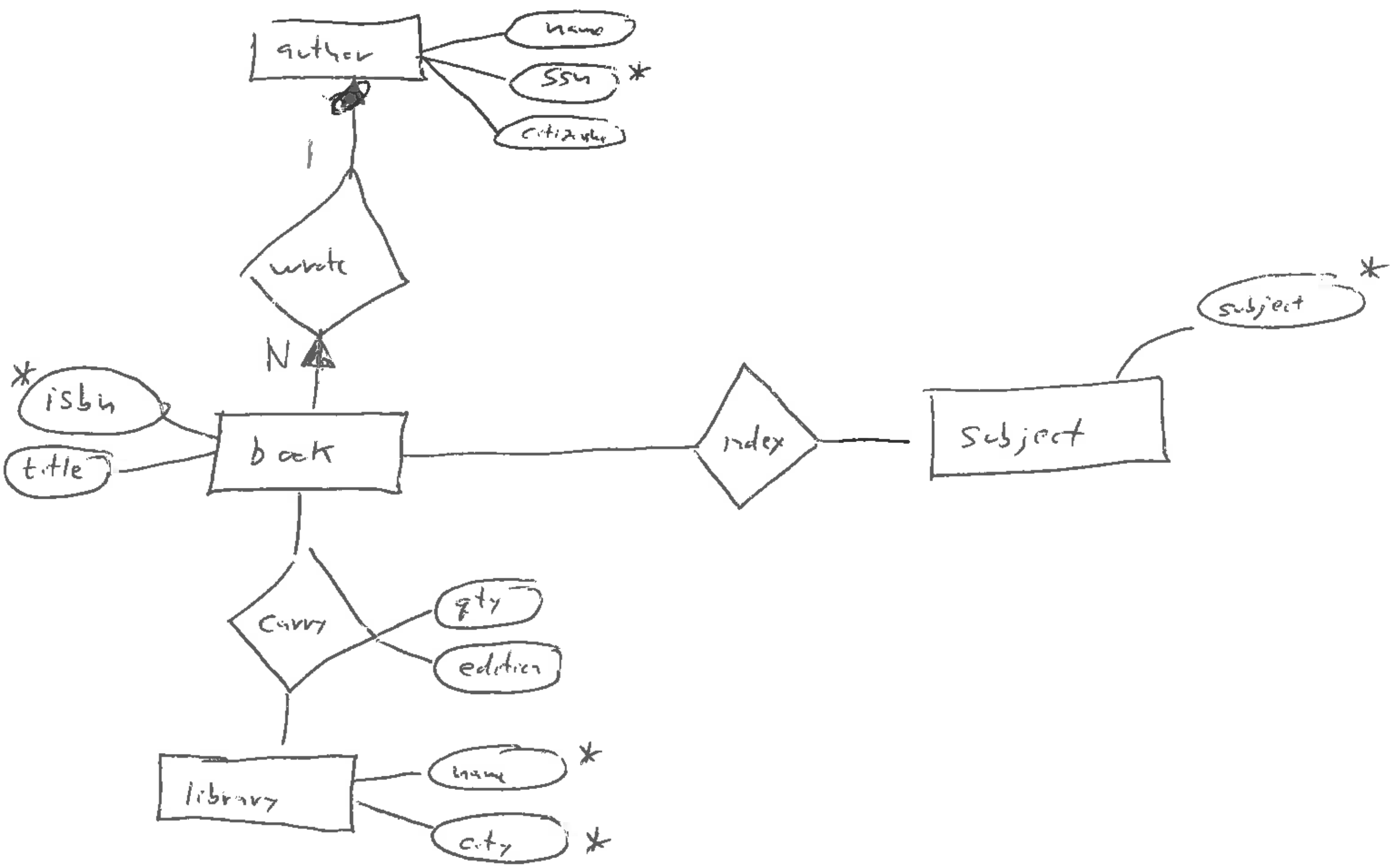


Figure 1: Library E-R model

Optimization: Functional relationships do not need a separate relation to be represented. For a functional relationship form set A to set B, incorporate the primary key of B into A.

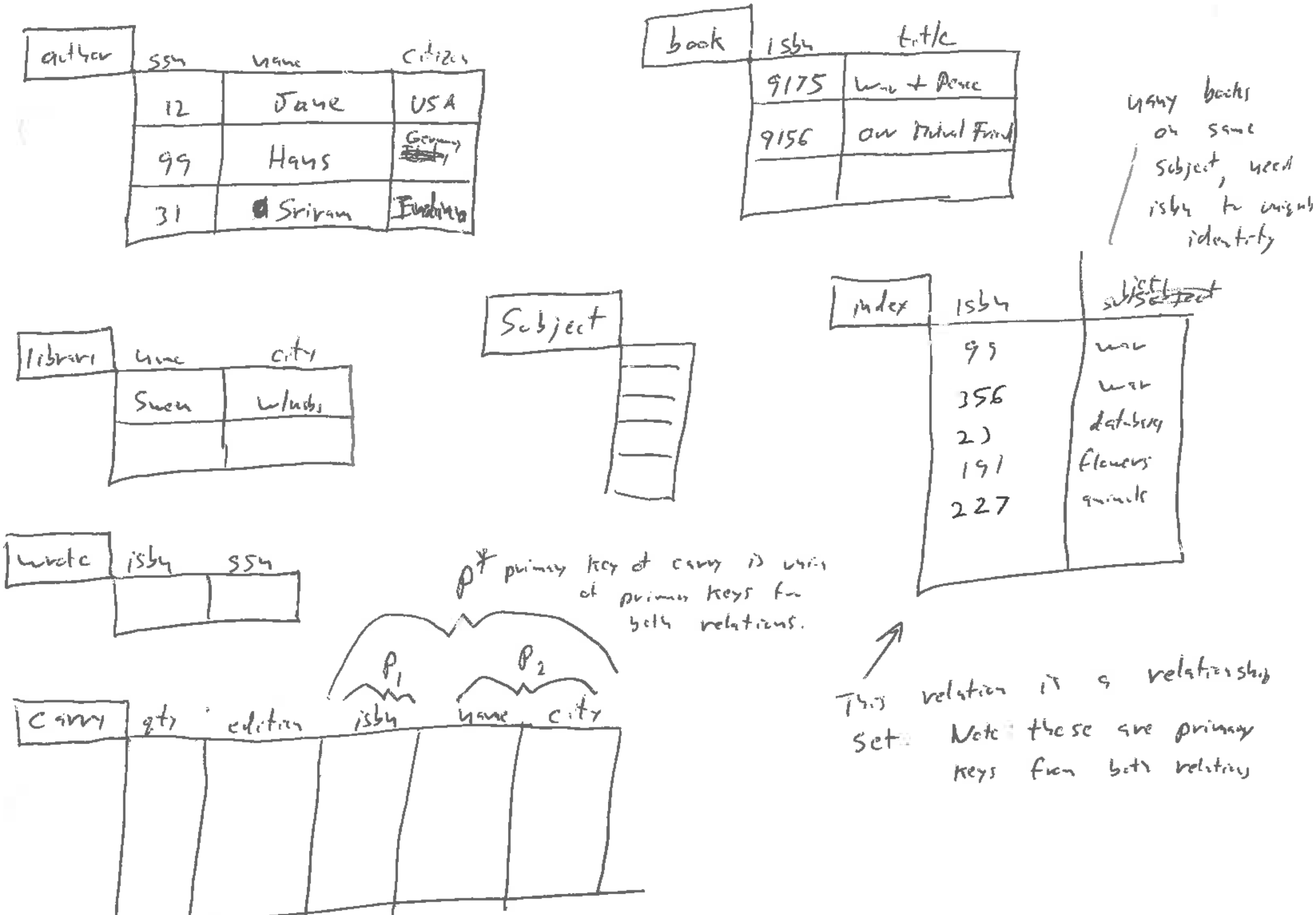


Figure 2: Library Relational Model

E.g. If every book has only one author, the *wrote* is a functional relationship. In the relational schema, get rid of the relation *wrote* and add ssn as an attribute of *book*. This save one relation.

General comments:

- A relation is a *set* of tuples. ~~No duplicates~~
- Order of tuples in a relation is not important.
- Order of attributes *is* important. To avoid this we give names to attributes.
- Every tuple is stored only once in a relation.
- A value may appear multiple times in a column. If a single attribute is the primary key for a relation, can a value appear multiple times in the column corresponding to that attribute?

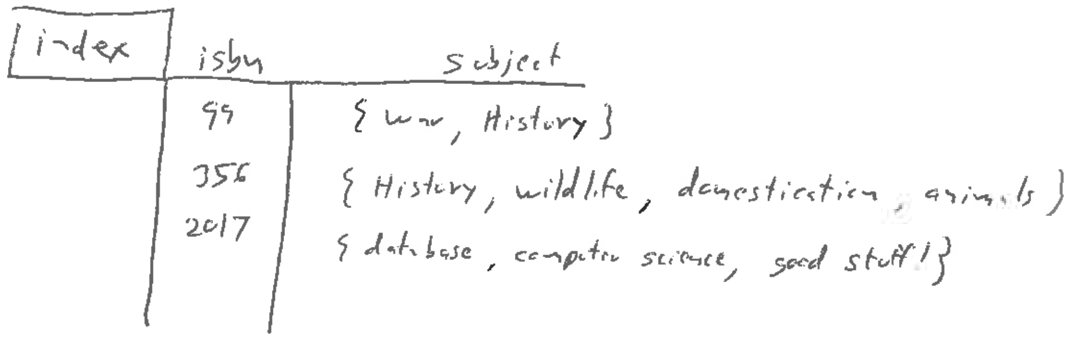


Figure 3: Set Model Approach

We *could* represent all relationships using set-valued attributes instead of atomic values:

Problems:

- non-standard width for fields
- long search time
- complicated DML (2nd order logic)

Relational model only allows fixed width fields.

Extensions to the Relational Model

- Referential Integrity: E.g. books must have an author.

Ex 2: Insert employee tok assuming works in toy dept. No toy dept in dept relation!

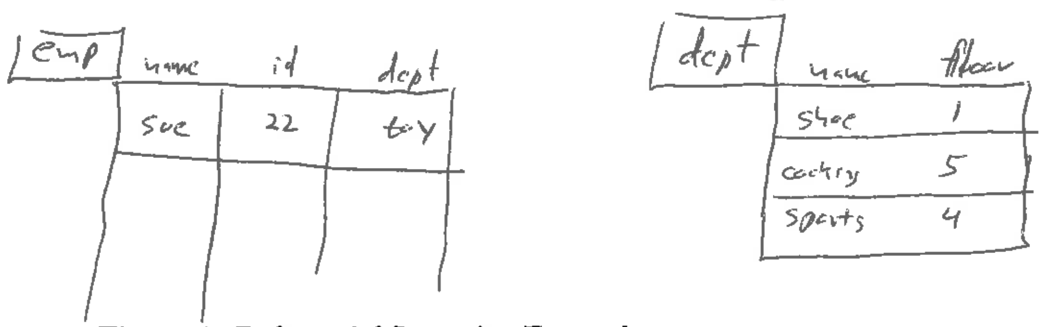


Figure 4: Referential Integrity Example

- Generalization Hierarchy: E.g my honda is a car, cars are vehicles, vehicles are physical objects.

- Order Relations: E.g. if some text is stored in a relation, word by word, there is an implicit order among the tuples.

Lecture 4: Relational Algebra

Query Languages are either:

- Procedural
- Nonprocedural

Relational Algebra is a formal procedural query language.

The language is composed of five fundamental operators:

$$\{\sigma \;,\; \pi \;,\; \times \;,\; \cup \;,\; - \}$$

Assume the following schemas:

Student(name,sid,ssn,status)
Faculty(name,fid,ssn,salary)
Courses(name,credits,dept, cnum,fid)
Grades(sid,cnum,semester,grade) ← add fid (needed on 5.5)
Enrolled(sid,cnum) /fid)

* Put on board and save

student			
name	sid	ssn	status
Bob	99	121	4
Sue	19	934	3
John	12	813	3
Mary	73	395	4

faculty			
name	fid	ssn	salary
Leut	9	912	1
Nicol	7	812	81
Park	3	452	105

courses				
name	credits	dept	cnum	fid
database	3	cs	521	9
simulation	5	cs	426	7
architecture	3	cs	314	3

grades			
sid	cnum	semester	grade
99	314	92.1	A
99	426	92.2	B
73	521	92.3	A
73	426	92.3	A
73	314	92.3	A
12	521	93.1	C

enrolled	
sid	cnum
99	321
99	111
19	321
82	101
73	321

Selection (σ)

Selects tuples that satisfy given predicate:

$\sigma_{name="leut"}(faculty)$ — leut 9 912 1

$\sigma_{name="Bob"}(students)$ — Bob 99 121 4

$\sigma_{status>3}(students)$ — { Bob 99 121 4 / Mary 73 395 4 }

$\sigma_{credits>3 \wedge dept="cs"}(courses)$ – simulation 5 cs 426 7

$\sigma_{cnum>500}(enrolled)$ {}

$\sigma_{sid=ssn}(student)$ {}

Say only want *name* of seniors.

Projection (π)

Copies relation with only columns given as arguments:

$\pi_{name,sid}(student)$

Bob	99
Sue	19
John	12
Mary	73

$\pi_{salary}(faculty)$

1
81
105

Names of seniors:

$\pi_{name}(\sigma_{status>3}(students))$

Say now want names of students enrolled in 321. Must have operator that takes two relations, student and enrolled, as arguments.

Cartesian Product ($\times$).

Takes the cross product off two relations, i.e. each tuple in the second relation is concatenated to each tuple in the first relation.

r = student $\times$ enrolled

Schema of resultant relation is all the attributes of both relations:

schema of **r** = (student.name, student.sid, student.ssn, student.status, enrolled.sid, enrolled.cnum) ;

How get names of all students enrolled in 321?

$\pi_{student.name}(\sigma_{enrolled.cnum=321}(students \times enrolled))$

Is this correct? **NO**

$student \times enrolled$:

student.name	student.sid	student.ssn	student.status	enrolled.sid	enrolled.cnum
Bob	99	121	4	99	321
Bob	99	121	4	99	111
Bob	99	121	4	19	321
Bob	99	121	4	82	101
Bob	99	121	4	73	321
Sue	19	934	3	99	321
Sue	19	934	3	99	111
Sue	19	934	3	19	321
Sue	19	934	3	82	101
Sue	19	934	3	73	321
John	12	813	3	99	321
John	12	813	3	99	111
John	12	813	3	19	321
John	12	813	3	82	101
John	12	813	3	73	321
Mary	73	395	4	99	321
Mary	73	395	4	99	111
Mary	73	395	4	19	321
Mary	73	395	4	82	101
Mary	73	395	4	73	321

Result of query would be Bob,Bob,Bob,Sue,Sue,Sue,John,John,John,Mary,Mary,Mary

Since relations are sets, and hence duplicates are removed, result is Bob,Sue,John, Mary. Note, John is **not** enrolled in 321.

How correct the query?

$$\pi_{student.name}(\sigma_{student.sid=enrolled.sid}(\sigma_{enrolled.cnum=321}(students \times enrolled))$$

How find name of all faculty who are teaching graduate courses?

$$\pi_{faculty.name}(\sigma_{faculty.fid=courses.fid}(\sigma_{courses.cnum>500}(faculty \times courses))$$

fac.name	fac.fid	fac.ssn	fac.salary	c.name	c.credits	c.dept	c.cnum	c.fid
Leut	9	912	1	database	3	cs	521	9
Leut	9	912	1	simulation	5	cs	426	7
Leut	9	912	1	architecture	3	cs	314	3
Nicol	7	812	81	database	3	cs	521	9
Nicol	7	812	81	simulation	5	cs	426	7
Nicol	7	812	81	architecture	3	cs	314	3
Park	3	452	105	database	3	cs	521	9
Park	3	452	105	simulation	5	cs	426	7
Park	3	452	105	architecture	3	cs	314	3

How get names of all students who have failed at least one course?

$\pi_{student.name}(\sigma_{student.sid=grades.sid}(\sigma_{grades.grade=F}(student \times grades))$

Suppose want to know all students who are taking 321 or 444?

Union ($\bigcup$)

$r1 \bigcup r2$ is the union of relations $r1$ and $r2$.

$r1 = \pi_{student.name}(\sigma_{student.sid=enrolled.sid}(\sigma_{erolled.cnum=321}(student \times enrolled)))$

$r2 = \pi_{student.name}(\sigma_{student.sid=enrolled.sid}(\sigma_{erolled.cnum=444}(student \times enrolled)))$

Answer: $r1 \bigcup r2$

Set Difference ($-$)

$r1 - r2$ is those members of $r1$ that are *not* in $r2$.

Say want students who are taking 321 but not 444:

$r1 - r2$.