COMP9311:
Database Systems

Term 2 2019

Week 3 Monday (Integrity Constraints, ER to Schema Mapping)

By Helen Paik, CSE UNSW

Textbook: Chapters 5, 6, 9

Disclaimer: the course materials are sourced from

- previous offerings of COMP9311 and COMP3311
- Prof. Werner Nutt on Introduction to Database Systems (http://www.inf.unibz.it/~nutt/Teaching/IDBs1011/)

Integrity Constraints

Ideal: DB instance reflects the real world

In real life: This is not always the case

Goal: Find out, when DB is out of sync

Observation: Not all mathematically possible instances make sense

- To represent real-world problems, need to describe
  - » what values are/are not allowed
  - » what combinations of values are/are not allowed

Idea:

- Formulate conditions that hold for all plausible instances
- Check whether the condition holds after an update
- Such conditions are called integrity constraints!

Common Types of Integrity Constraints

Functional Dependencies (FDs)

- “Employees in the same department have the same boss”

Superkeys and keys (special case of FDs)

- “Employees with the same tax code are identical”

Referential Integrity (also “foreign key constraints”)

- “Employees can only belong to a department that is mentioned in the Department relation”

Domain Constraints

- “No employee is younger than 15 or older than 80”

Integrity constraints are part of the schema. We allow only instances that satisfy the integrity constraints

Functional Dependencies (Example)

Emp (Name, taxCode, Dept, DeptHead)

A state of Emp that contains two tuples with

- the same Dept, but different DeptHead is wrong …
- the same taxCode, but different Name, Dept, or DeptHead is also wrong …

We write the **desired conditions** symbolically as

- Dept $\rightarrow$ DeptHead
- taxCode $\rightarrow$ Name, Dept, DeptHead.

We read:

- "Dept functionally **determines** DeptHead", or
- "Name, Dept, and DeptHead *functionally depend* on taxCode"

Functional Dependencies

A functional dependency on R is an expression

$$A_1,\ldots,A_m \rightarrow B_1, \ldots,B_n$$

where $A_1,\ldots,A_m$ and $B_1, \ldots,B_n$ are attributes of R.

An instance r of R satisfies the FD if for all tuples t_1, t_2 in R

$$t_1[A_1,\ldots,A_m] = t_2[A_1,\ldots,A_m]$$

implies

$$t_1[B_1, \ldots,B_n] = t_2[B_1, \ldots,B_n]$$

FDs: Example

Which FDs does this instance satisfy?

- EmpID → Name, Phone, Position
- Position → Phone
- Phone → Position

Emp (EmpID, Name, Phone, Position)

with instance

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	Jones	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Brown	1234	Lawyer

General Approach for Checking FDs

To check $A \rightarrow B$ on an instance,

- erase all other columns

…	A	…	B	
	X1		Y1	
	X2		Y2	
	…		…	

- check if the remaining relation is **functional** in A

FDs, Superkeys and Keys

Consider that the following FDs hold

Person (SSN, Name, DOB)
SSN → Name, DOB

Product (Name, Price, Manufacturer)
Name → Price, Manufacturer
Name, Price → Name, Price, Manufacturer

Book (Author, Title, Edition, Price)
Author, Title, Edition → Price

- A set of attributes of a relation is a superkey if it functionally determines all the attributes of the relation

- A superkey is a candidate key if none of its subsets is a superkey

i.e., Candidate keys are minimal superkeys

Keys: Definitions

Superkey

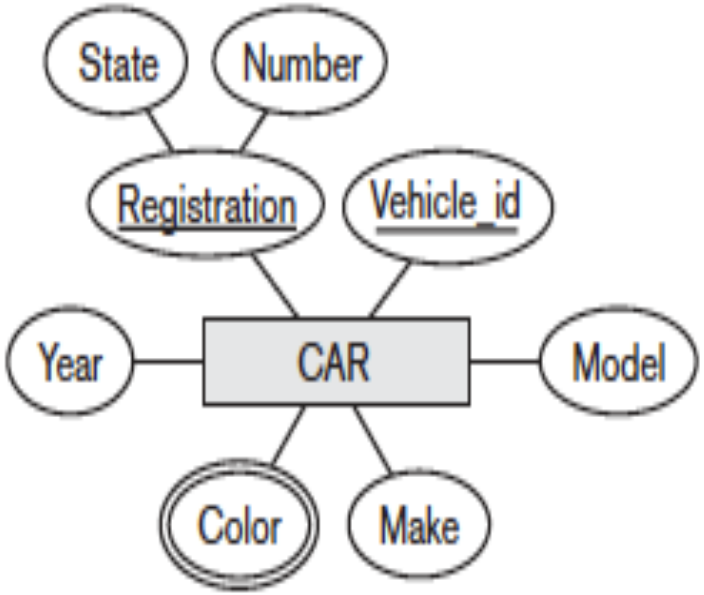
- a set of attributes whose values together uniquely identify a tuple in a relation

Candidate Key

- a superkey for which no proper subset is a superkey:
- a superkey that is minimal
- can be more than one for a relation

Primary Key

- a candidate key chosen to be the main key
- one for each relation, indicated by underlining the key attributes



Car(Registration, Vehicle_ID, Year, Model, Colour, Make)

Example: Multiple Keys

Student (Lastname, Firstname, MatriculationNo, Major)

Candidate key
(2 attributes)

Candidate key

Superkey

Note: There are alternate candidate keys

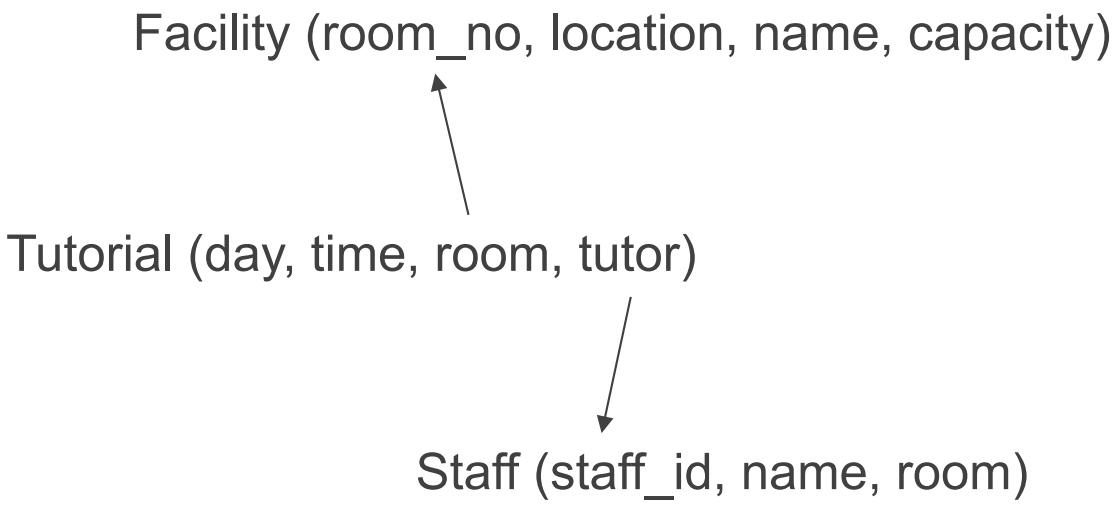
Candidate keys are

{Lastname, Firstname} and {MatriculationNo}

Primary key is? (note: most of the time, a primary key is a single attribute)

Foreign Keys

- A set of attributes in a relation that exactly matches the primary key in another relation
- the names of the attributes don't have to be the same but must be of the same domain



Notation:

FK1: Tutorial (room) references Facility (room_no)
FK2: Tutorial (tutor) references Staff (staff_id)

Satisfaction of Foreign Key Constraints

"FK: R(A) references S(B)" is satisfied by an instance of R and S if for every t_1 in R there is a t_2 in S such that $t_1[A] = t_2[B]$, provided $t_1[A]$ is not null

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Foreign key constraints are also called "referential integrity constraints."

Updates May Violate Constraints …

Updates are

Insertions, Deletions, Modifications

of tuples

Example DB with tables as before:

Student (studno, name, hons, tutor, year)

Staff (lecturer, roomno, appraiser)

FK: Student(tutor) references Staff(lecturer)

The DB has key and foreign key constraints … what can go wrong? How should the DBMS react?

Insertions (1)

If the following tuple is inserted into Student, what should happen? Why?

(s1, jones, cis, capon, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Insertions (2)

If the following tuple is inserted into Student, what should happen? Why?

(null, jones, cis, capon, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Insertions (3)

If the following tuple is inserted into Student, what should happen? Why?

(s7, jones, cis, null, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Insertions (4)

If the following tuple is inserted into Student, what should happen? Why?

(s7, jones, cis, calvanese, 3)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Deletions (1)

If the following tuple is deleted from Student, is there a problem? And what should happen?

(s2, brown, cis, kahn, 2)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Deletions (2)

And if this one is deleted from Staff ?

(kahn, IT206, watson)

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (1)

What if we change in Student

(s1, jones, ca, bush, 2)

to

(s1, jones, ca, watson, 2) ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (2)

And what if we change in Student

(s2, brown, cis, kahn, 2)

to

(s1, jones, ca, bloggs, 2) ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (3)

And what if we change in Staff

(lindsey, 2.10, woods)

to

(lindsay, 2.10, woods) ?

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Modifications (4)

Now, let's change in Staff

(goble, 2.82, capon)

to

(gobel, 2.82, capon) …

Student

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

Staff

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

Summary - Relational Database System

A *relational database schema* is

- a set of relation schemas $\{ R_1, \ R_2, \ \ldots \ R_n \}$, and
- a set of integrity constraints

A *relational database instance* is:

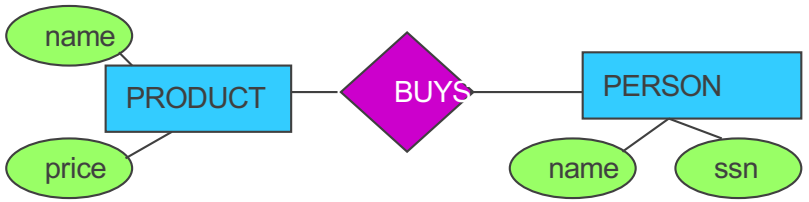
- a set of relation instances $\{ r_1(R_1), \ r_2(R_2), \ \ldots \ r_n(R_n) \}$
- where all of the integrity constraints are satisfied

One of the important functions of a relational DBMS:

- ensure that all data in the database satisfies constraints
- Integrity constraints: Domain cs, FDs, Keys, FKs
- Updates may violate ICs — and the DMBS has to react

Mapping ER Diagram to Relational Schema

Conceptual Model:



Relational Model:

We cannot store date in an ER schema

➔ We have to translate our ER schema into a relational schema

➔ What does “translation” mean?

Ideally, the mapping should

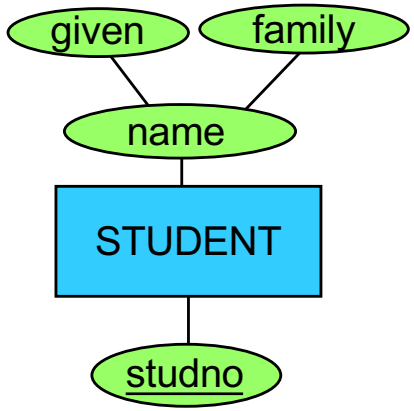
- be one-to-one in both directions
- not lose any information

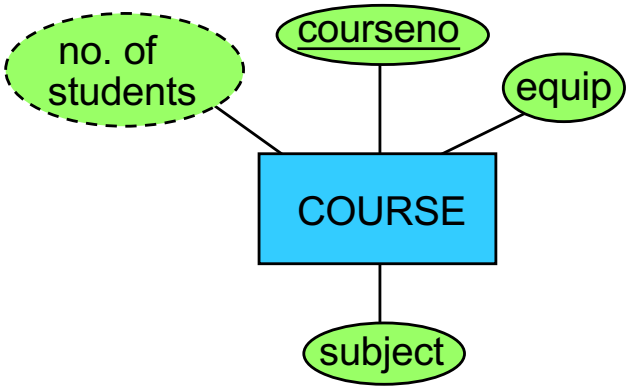
Mapping Entity Types to Relations

General rules:

- for every entity type create a relation
- every atomic attribute of the entity type becomes a relation attribute
  - composite attributes: include all the atomic attributes
  - derived attributes are not included
    (but remember their derivation rules)

Attributes of the entity key make up the primary key of the relation





Mapping Many:many Relationship Types

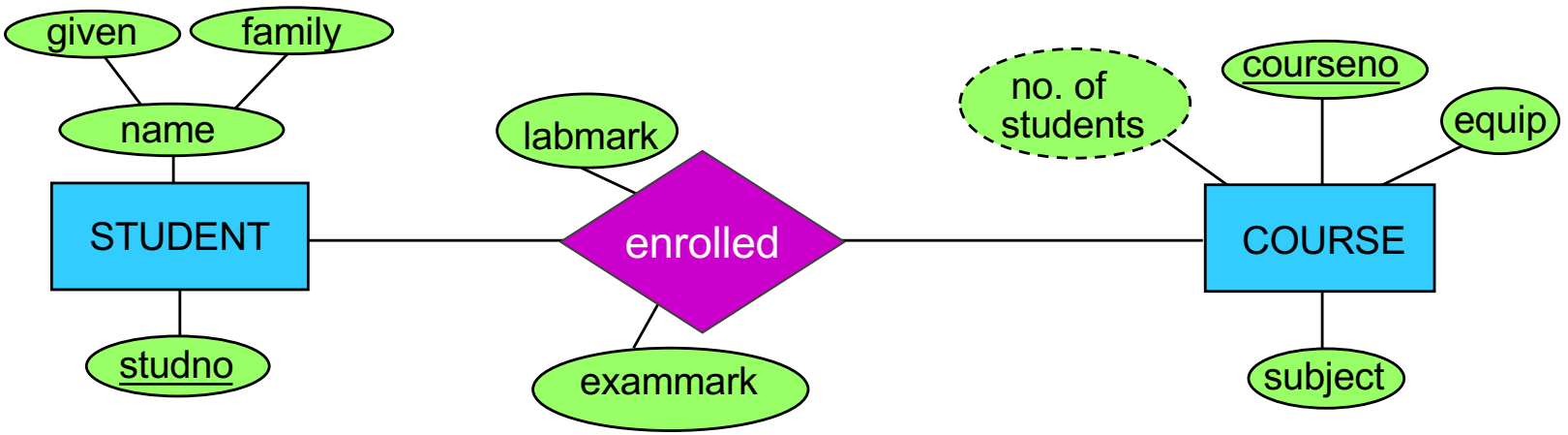
Rule: Create a relation with the following set of attributes:

$$\bigcup_{i=1}^{N} \text{primary\_key}(E_i) \cup \{a_1,\ldots,a_M\}$$

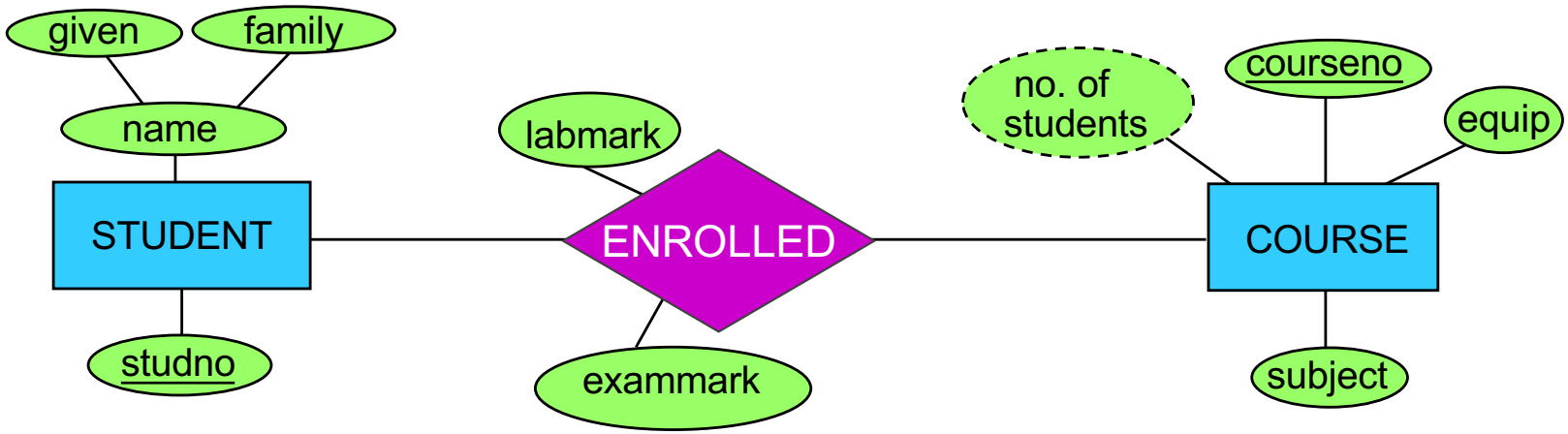
N *(degree of relationship)*

primary keys of each entity type participating in the relationship

attributes of the relationship type (if any)



Mapping Many:many Relationship Types



ENROL(studno, courseno, labmark, exammark)

Foreign Key ENROL(studno) references STUDENT(studno)

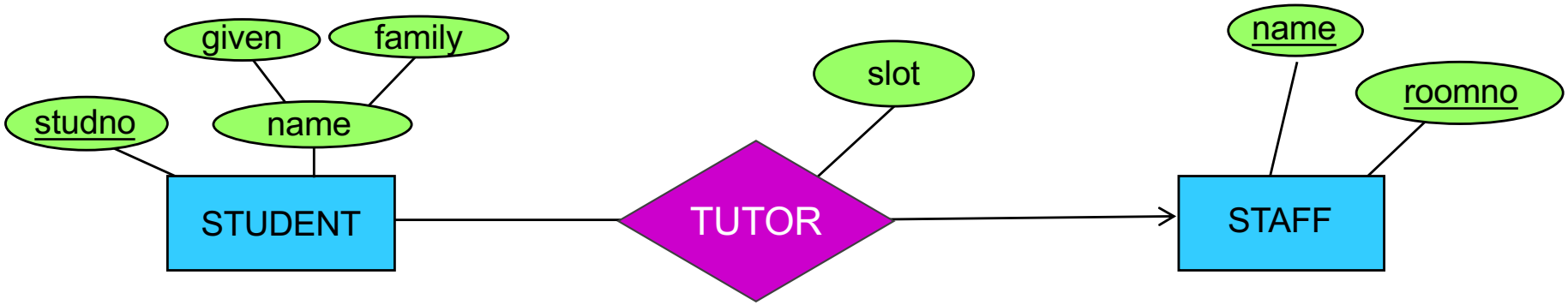
Foreign Key ENROL(courseno) references COURSE(courseno)

Mapping Many:One Relationship Types

Idea: "Post the primary key"

Rule: given E1 at the 'many' end of relationship and E2 at the 'one' end of the relationship, add information to the relation for E1

The primary key of the entity at the 'one' end (the determined entity) becomes a foreign key in the entity at the 'many' end (the determining entity). Include any relationship attributes with the foreign key entity



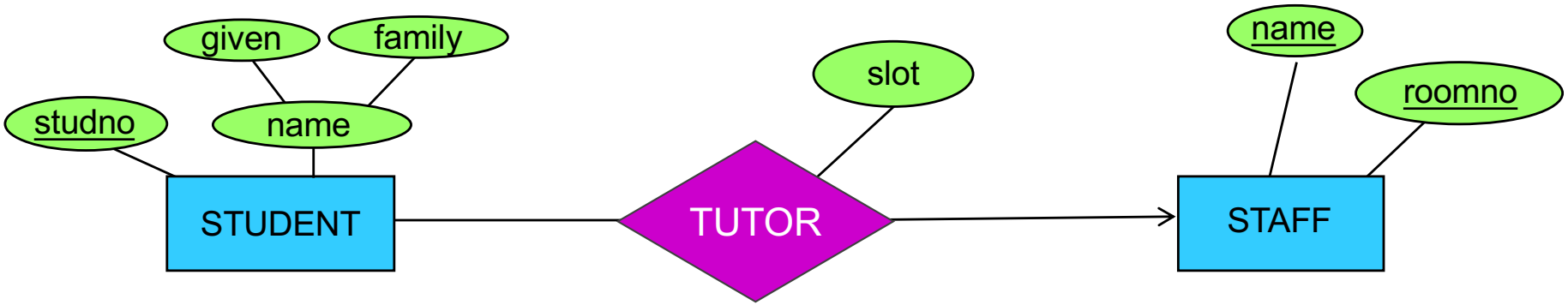
$$E1 \;\cup\; primary\_key(E2) \;\cup\; \{a_1,\ldots,a_n\}$$

relation for entity E1

primary key for E2, *is now a* foreign key *to* E2

attributes on the relationship (if any)

Mapping Many:one Relationship Types



The relation

STUDENT(studno, givenname, familyname)

is extended to

STUDENT(studno, givenname, familyname, tutor, roomno, slot)

and the constraint

Foreign Key STUDENT(tutor,roomno) references STAFF(name,roomno)

Mapping Many:one Relationship Types

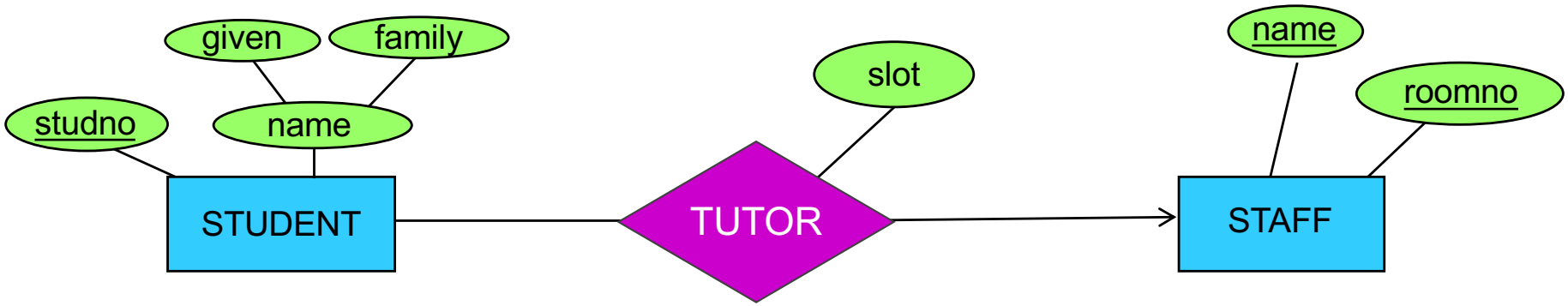
STUDENT

studno	given	family	tutor	roomno	slot
s1	fred	jones	bush	2.26	12B
s2	mary	brown	kahn	IT206	12B
s3	sue	smith	goble	2.82	10A
s4	fred	bloggs	goble	2.82	11A
s5	peter	jones	zobel	2.34	13B
s6	jill	peters	kahn	IT206	12A

STAFF

name	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

Mapping Many:one Relationship Types



Another Idea: If

- the relationship type is *optional* to both entity types, and
- an instance of the relationship is *rare*, and
- there are *many attributes* on the relationship then…

… create a new relation with the set of attributes:

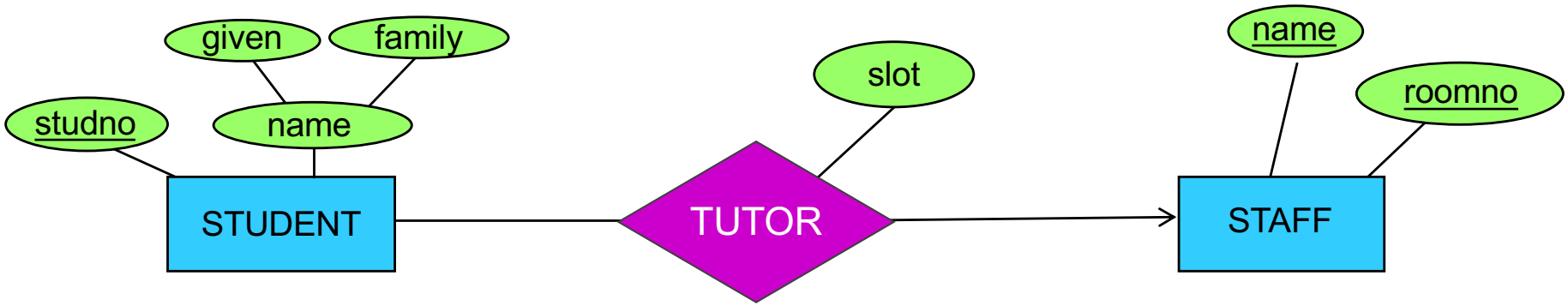
$$\text{primary\_key}(E_1) \cup \text{primary\_key}(E_2) \cup \{a_1,\ldots,a_m\}$$

primary key for E1, *is now a* foreign key *to* E1*; also the PK for this relation*

primary key for E2, *is now a* foreign key *to* E2

Any attributes on the relationship type

Mapping Many:one Relationship Types



Compare this with the mapping of M:M types!

TUTOR(studno, staffname, rommno, slot)

Foreign key TUTOR(studno) references STUDENT(studno)

Foreign key TUTOR(staffname, roomno) references STAFF(name, roomno)

Mapping Many:one Relationship Types

STUDENT

studno	given	family
s1	fred	jones
s2	mary	brown
s3	sue	smith
s4	fred	bloggs
s5	peter	jones
s6	jill	peters

STAFF

name	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

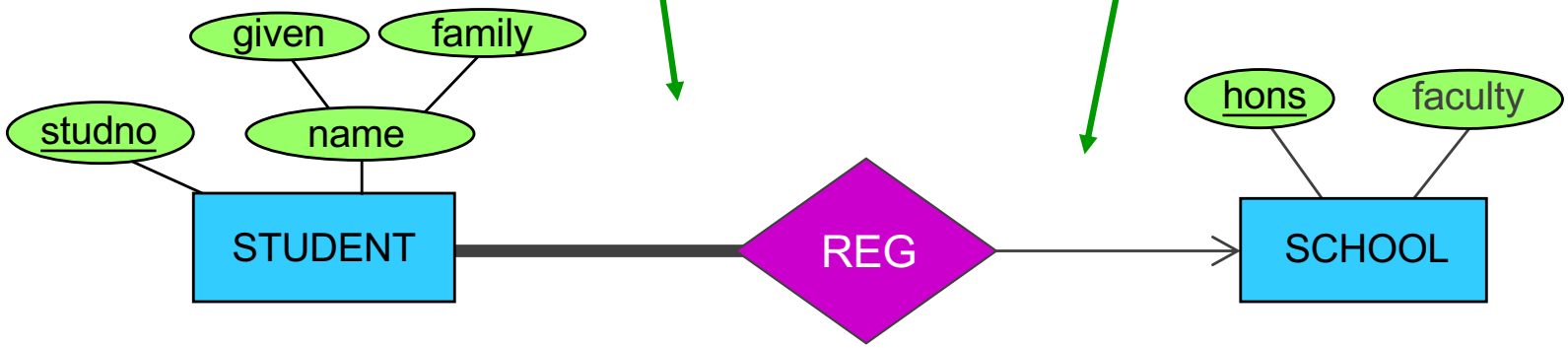
TUTOR

studno	tutor	roomno	slot
s1	bush	2.26	12B
s2	kahn	IT206	12B
s3	goble	2.82	10A
s4	goble	2.82	11A
s5	zobel	2.34	13B
s6	kahn	IT206	12A

Optional Participation of the Determined Entity ('one end')

A student entity instance must *participate in a relationship instance of* REG

A school entity instance need not *participate in a relationship instance of* REG



SCHOOL (hons, faculty)

STUDENT (studno, givenname, familyname, ???)

Optional Participation of the Determined Entity ('one end')

STUDENT

studno	given	family	hons
s1	fred	jones	ca
s2	mary	brown	cis
s3	sue	smith	cs
s4	fred	bloggs	ca
s5	peter	jones	cs
s6	jill	peters	ca

SCHOOL

hons	faculty
ac	accountancy
is	information systems
cs	computer science
ce	computer science
mi	medicine
ma	mathematics

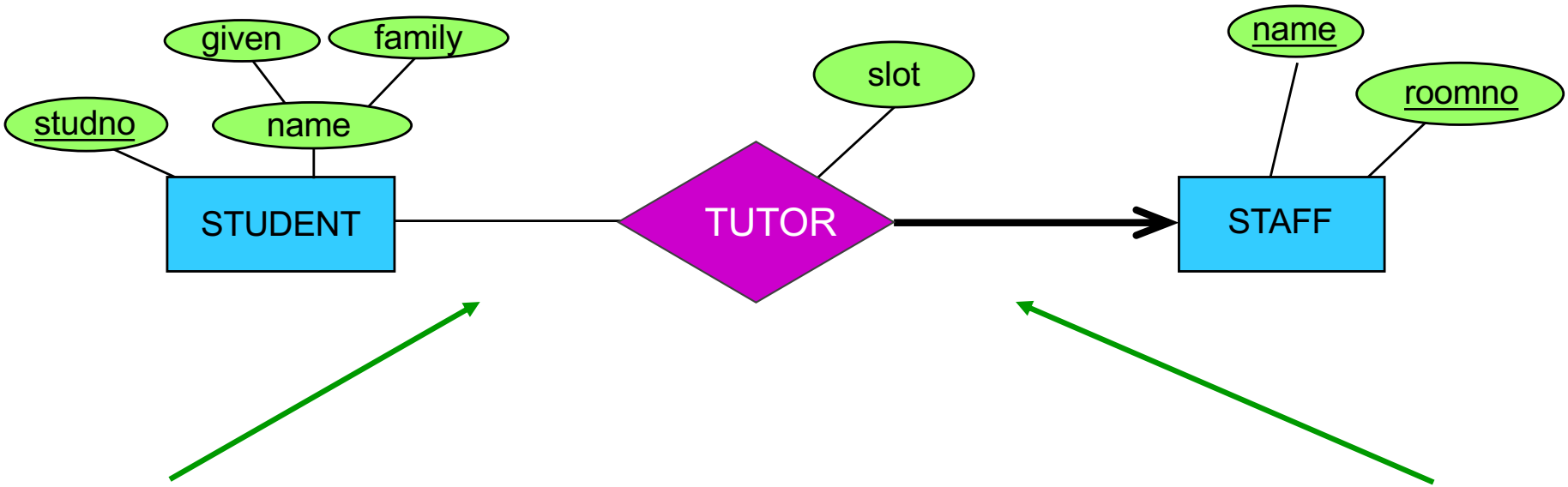
"hons" cannot be NULL because it is mandatory for a student to be registered for a school

➔ "not null" constraint

No student is registered for "mi", so "mi" doesn't occur as a foreign key value

(but that's no problem)

Optional Participation of the Determinant Entity ('many end')



A student entity instance need not *participate in a relationship instance of* TUTOR

A staff entity instance must *participate in a relationship instance of* TUTOR

Optional Participation of the Determinant Entity ('Many end')

CASE 1:

STUDENT (studno, givenname, familyname, tutor, roomno, slot)

STAFF(name, roomno)

CASE 2:

STUDENT(studno, givenname, familyname)

STAFF(name, roomno)

TUTOR(studno, tutor, roomno, slot)

Optional Participation of the Determinant Entity ('Many end')

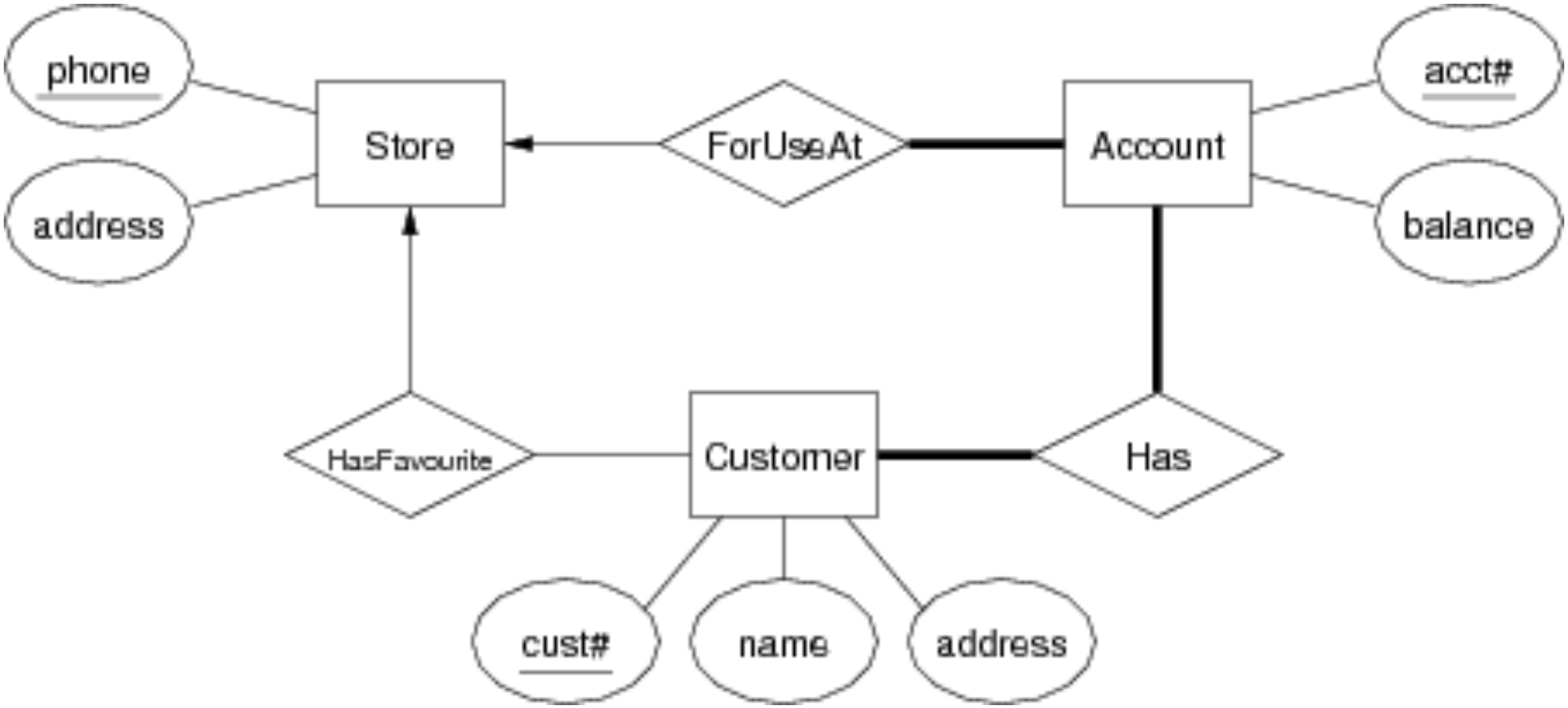
STUDENT

studno	given	family	tutor	roomno	slot
s1	fred	jones	bush	2.26	12B
s2	mary	brown	kahn	IT206	12B
s3	sue	smith	goble	2.82	10A
s4	fred	bloggs	goble	2.82	11A
s5	peter	jones	NULL	NULL	NULL
s6	jill	peters	kahn	IT206	12A

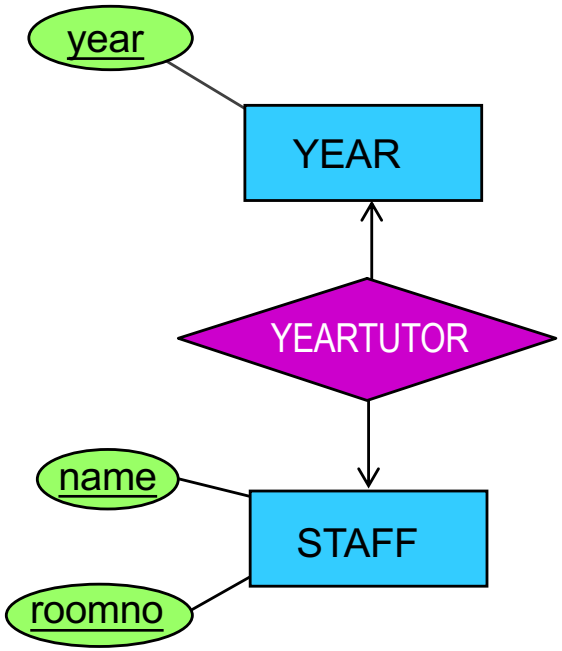
STAFF

name	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

Exercise 1


phone
address
Store
ForUseAt
Account
acct#
balance
HasFavourite
Customer
Has
cust#
name
address

Mapping One:one Relationship Types



Post the primary key of one of the entity types into the other entity type as a foreign key, including any relationship attributes with it

or

Merge the entity types together

YEAR

year	yeartutor
1	zobel
2	bush
3	capon

STAFF

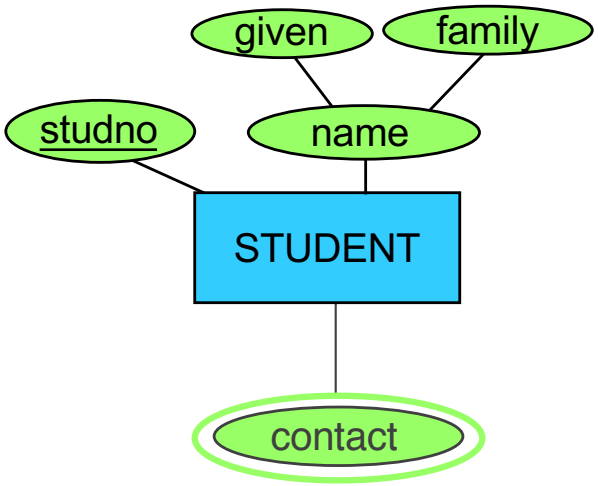
name	roomno	year
kahn	IT206	NULL
bush	2.26	2
goble	2.82	NULL
zobel	2.34	1
watson	IT212	NULL
woods	IT204	NULL
capon	A14	3
lindsey	2.10	NULL
barringer	2.125	NULL

Multi-Valued Attributes

For each multi-valued attribute of E_i, create a relation with the attributes

primary_key(E_i) U multi-valued attribute

The primary key comprises all attributes



STUDENT

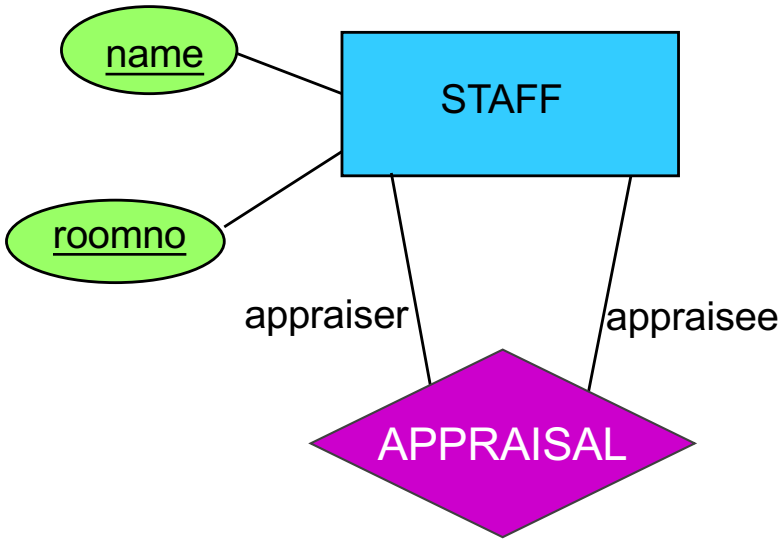
studno	given	family
s1	fred	jones
s2	mary	brown

STUDENT_CONTACT

studno	contact
s1	Mr. Jones
s1	Mrs Jones
s2	Bill Brown
s2	Mrs Jones
s2	Billy-Jo Woods

Mapping Roles and Recursive Relationships

How can the entity STAFF appear in both of its roles ?



STAFF(<u>name</u>, <u>roomno</u>, appraiser, approomno)