Database Systems and Web (15B11CI312)

Database Systems and Web

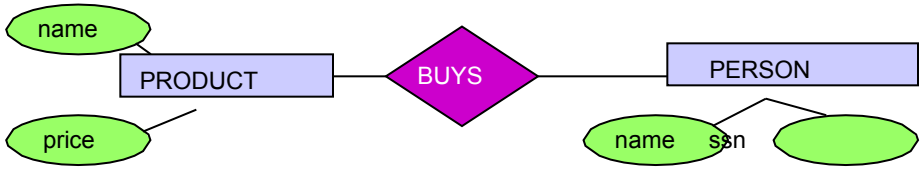
Lecture 7: ER to Relational Model

Contents to be covered

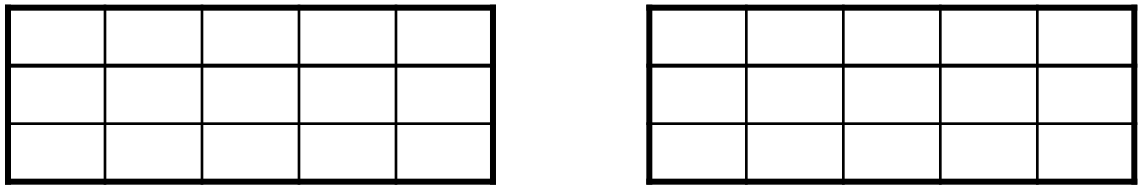
- Conceptual and Logical Design
- Translation Principles
- ER to Relational Table: Algorithm
- Mapping categories
- Demonstration with example

Conceptual and Logical Design

Conceptual Model:



Relational Model:



Mapping an E-R Diagram to a Relational Schema

We cannot store date in an ER schema
(there are no ER database management systems)

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

€ What does “translation” mean?

Translation: Principles

- Maps
  - ER schemas to relational schemas
  - ER instances to relational instances
- Ideally, the mapping should
  - be one-to-one in both directions
  - not lose any information
- Difficulties:
  - what to do with ER-instances that have identical attribute values, but consist of different entities?
  - in which way do we want to preserve information?

Mapping ER and EER Schemas into the Relational Model

Steps of The Algorithm

- STEP 1: Map Entity Types

– Each strong Entity to a table

-All simple attributes will become column in the table

-Include only simple attributes of the composite attribute in the table as columns

-Derived attribute will not become part of the table

-Choose key attribute as primary key of the table

- STEP 2: Map Weak Entity Types to a table and draw identifier from parent entity type into weak entity type

- Key of weak entity will be partial key of weak entity and key attribute of the owner entity on which it depends.

Map Relationship Types (STEP 3):

1:1 - options for setting up one, two or three relations

- Include PK of one of the entity T into other, say S, better to choose the PK of the entity type T and include that in the entity S with total participation in the relation.

- Include attributes of R in S
- No table for R
- Or a table for R with PK of both plus its own attributes or all the attributes into one relation

Map Relationship Types (STEP 3):

1:N – the many side of the relationship type T provides a PK to the one side, say S, no new relation

- include attributes of R into S

M:N – need to set up a separate relation for the relationship

- include PKs of T and S , and attributes of R into new table

STEP 4: Map multivalued attributes – set up a new relation for each multi-valued attribute and the PK of the corresponding entity type

STEP 5: Mapping of generalization hierarchies and set-subset relationships – possiblity of collapsing into one relation vs. as many relations as the number of distinct classes.

Convert each subclass S and superclass C, where attributes of C are {k, a,b..} and k is PK of C into a relation using following

1. Create a table L for C with attributes of L are {k, a, b..} and PK(L) = k. Create a table for each subclass S, with attributes of S are {k} U attributes of S, with PK of S as {k}
2. Create a relation for S , with attributes of S as {k,a, b..} and its own attributes and PK = k.

Mapping Categories

Specify a new key called a surrogate key when creating a relation for category.

(Because keys for all participating classes are different)

Include any attribute of its own

Add the surrogate key as foreign key to all other participating relations

If a category's superclasses share the same key , there is no need for surrogate key

Demonstration

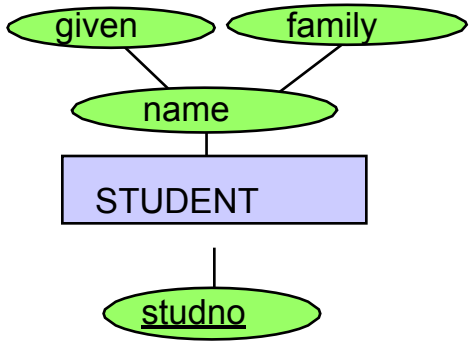
THEORY WITH EXAMPLE

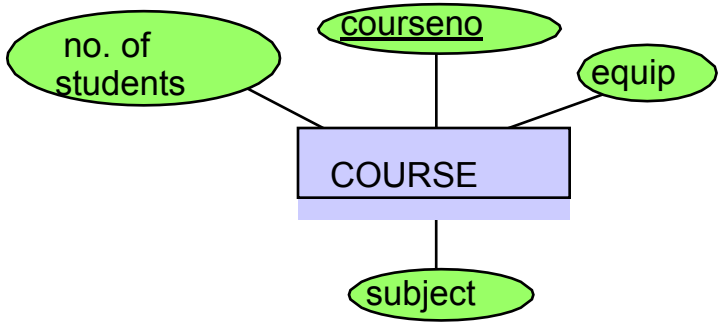
Mapping Entity Types to Relations

- 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

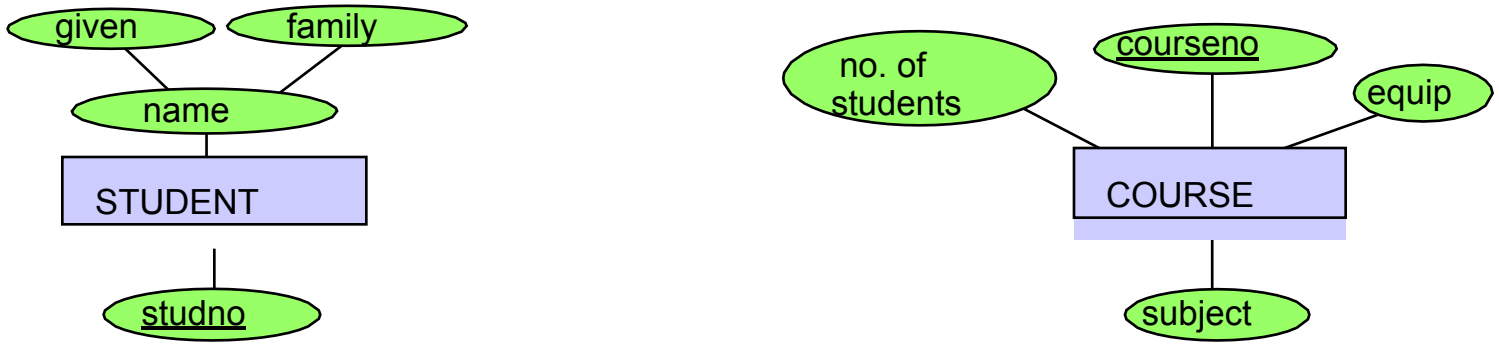
(but remember their *derivation rules*)

- Relation instances are subsets of the cross product of the domains of the attributes
- Attributes of the *entity key* make up the *primary key* of the relation





Mapping Entity Types to Relations (contd..)



STUDENT (studno, givenname, familyname)

COURSE (courseno, subject, equip)

Mapping M:N Relationship Types to Relations

Create a relation with the following set of attributes:

N *(degree of relationship)*

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

primary keys of each entity type participating in the relationship

attributes of the relationship type (if any)

given family

name

STUDENT

studno

labmark

exammark

no. of students

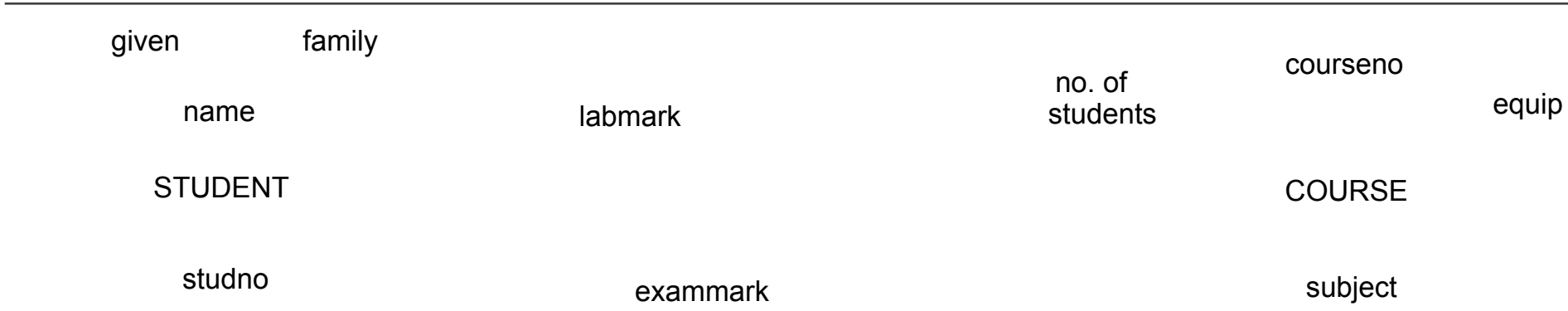
courseno

COURSE

subject

equip

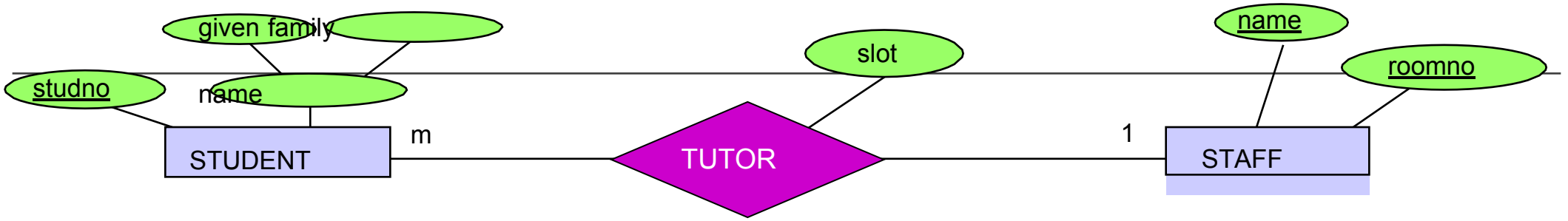
Mapping M:N Relationship Types to Relations (contd..)



ENROL(studno, courseno, labmark, exammark)
Foreign Key ENROL(studno) references STUDENT(studno) Foreign Key

ENROL(courseno) references COURSE(courseno)

Mapping M:1 Relationship Types to Relations



Idea: *"Post the primary key"*

- 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

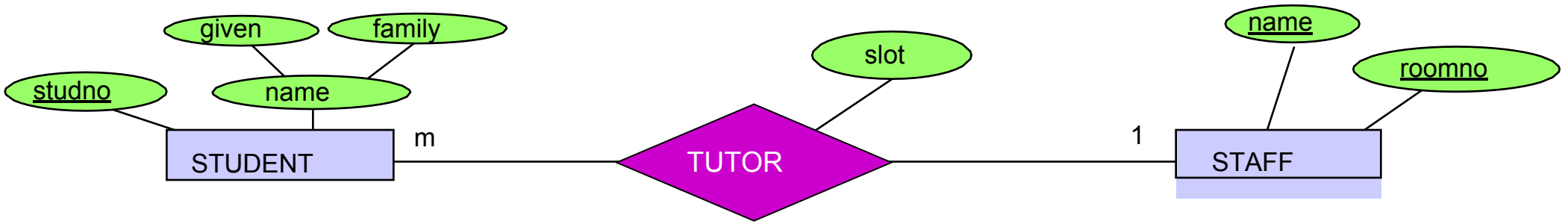
$$E1 \cup primary\_key(E2) \cup \{a_1,\ldots,a_n\}$$

relation for entity E1

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

attributes on the relationship type (if any)

Mapping M:1 Relationship Types to Relations: Example



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 M:1 Relationship Types to Relations (contd..)

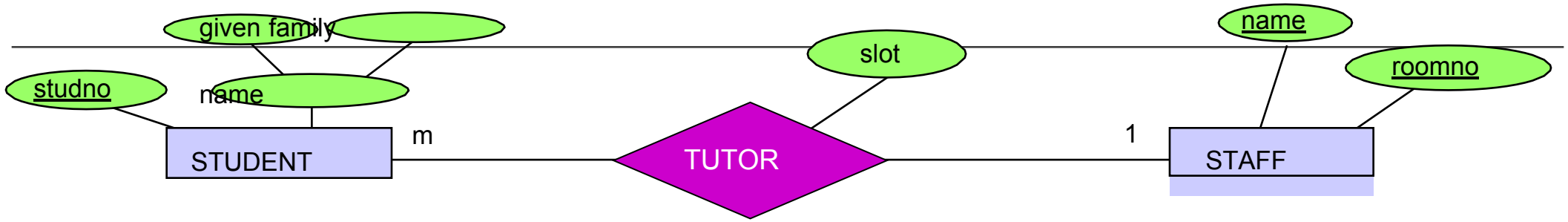
STUDENT

studno	given	family	tutor	roomno	slot
s1	prakash	tyagi	T1	2.26	12B
s2	mukesh	sharma	T2	IT206	12B
s3	suresh	singh	T3	2.82	10A
s4	prakash	verma	T3	2.82	11A
s5	pawan	tyagi	T4	2.34	13B
s6	bhuwan	nigam	T2	IT206	12A

STAFF

name	roomno
T2	IT206
T1	2.26
T3	2.82
T4	2.34
T5	IT212
T6	IT204
T7	A14
T8	2.10
T9	2.125

Mapping M:1 Relationship Types to Relations (contd..)



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:

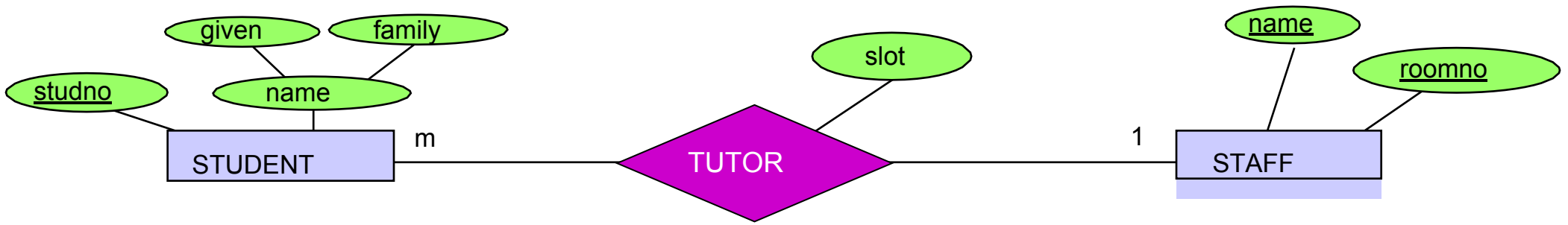
primary_key(E1) U primary_key(E2) U $\{a_1,\ldots,a_m\}$

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

attributes on the relationship type (if any)

Mapping M:1 Relationship Types to Relations (contd..)



Compare with the mapping of M:N relationship types!

TUTOR(studno, staffname, rommno, slot)

and

Foreign key TUTOR(studno) references STUDENT(studno)

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

Mapping M:1 Relationship Types to Relations (contd..)

STUDENT

studno	given	family
s1	prakash	tyagi
s2	mukesh	sharma
s3	suresh	singh
s4	prakash	verma
s5	pawan	tyagi
s6	bhuwan	

STAFF

name	roomno
T2	IT206
T1	2.26
T3	2.82
T4	2.34
T5	IT212
T6	IT204
T7	A14
T8	2.10
T9	2.125

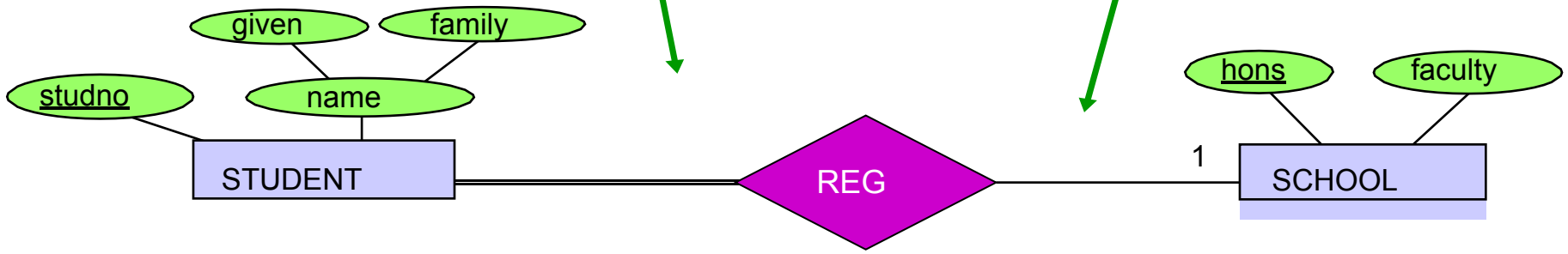
TUTOR

studno	tutor	roomno	slot
s1	T1	2.26	12B
s2	T2	IT206	12B
s3	T3	2.82	10A
s4	T3	2.82	11A
s5	T4	2.34	13B
s6	T2	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 (<u>hons</u>, faculty)

STUDENT (<u>studno</u>, givenname, familyname, ???)

Optional Participation of the Determined Entity

STUDENT

studno	given	family	hons
s1	prakash	tyagi	ca
s2	mukesh	sharma	cis
s3	suresh	singh	cs
s4	prakash	verma	ca
s5	pawan	tyagi	ce
	huwan	nigam	ma

SCHOOL

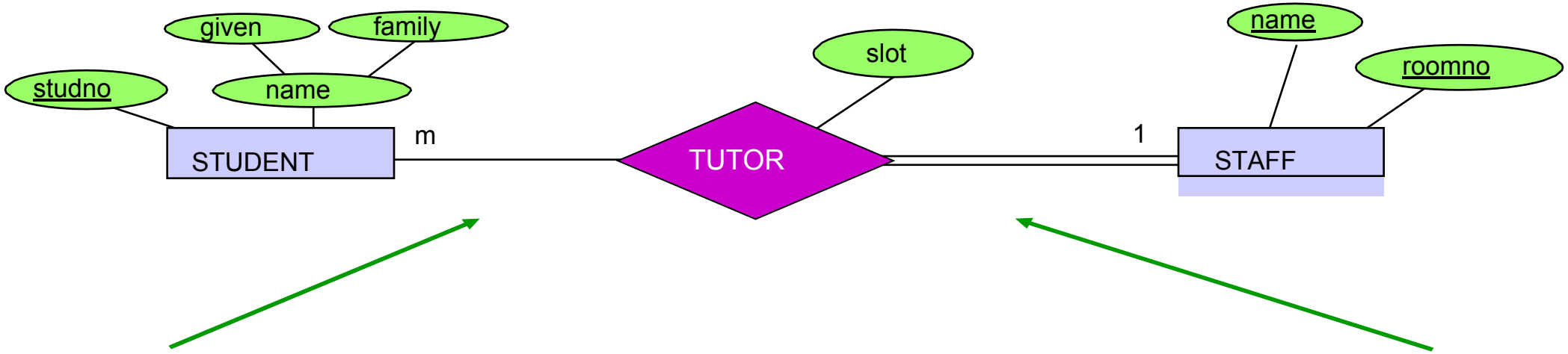
hons	faculty
ca	accountancy
cis	information systems
cs	computer science
ce	computer science
mi	medicine
ma	mathematics

"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

1. STUDENT (<u>studno</u>, givenname, familyname, tutor, roomno, slot)

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

   Integrity constraint:

$$\pi_{name,roomno} \text{STAFF} \setminus \pi_{tutor,roomno} \text{STUDENT} = \varnothing$$

2. STUDENT(<u>studno</u>, givenname, familyname) STAFF(<u>name</u>, <u>roomno</u>)

   TUTOR(<u>studno</u>, tutor, roomno, slot)

   Do we also need an integrity constraint?

Optional Participation of the Determinant Entity (contd..)

STUDENT

studno	given	family	tutor	roomno	slot
s1	prakash	tyagi	T1	2.26	12B
s2	mukesh	sharma	T2	IT206	12B
s3	suresh	singh	T3	2.82	10A
s4	prakash	verma	T3	2.82	11A
s5	pawan	tyagi	T4	2.34	13B
s6	bhuwan	nigam	T2	IT206	12A

STAFF

name	roomno
T2	IT206
T1	2.26
T3	2.82
T4	2.34
T5	IT212
T6	IT204
T7	A14
T8	2.10
T9	2.125

Optional Participation of the Determinant Entity (contd..)

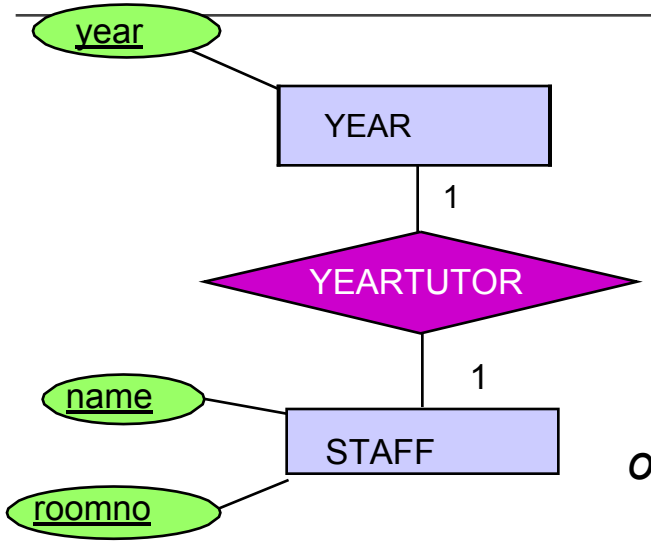
STUDENT

studno	given	family	tutor	roomno	slot
s1	prakash	tyagi	T1	2.26	12B
s2	mukesh	sharma	T2	IT206	12B
s3	suresh	singh	T3	2.82	10A
s4	prakash	verma	T3	2.82	11A
s5	pawan	tyagi	NULL	NULL	NULL
s6	bhuwan	nigam	T2	IT206	12A

STAFF

name	roomno
T2	IT206
T1	2.26
T3	2.82
T4	2.34
T5	IT212
T6	IT204
T7	A14
T8	2.10
T9	2.125

Mapping 1:1 Relationship Types to Relations



- 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

Which constraint holds in this case?

YEAR

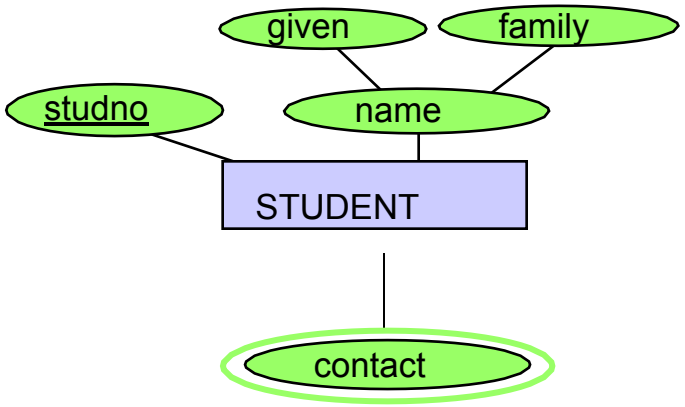
year	yeartutor
1	T4
2	T1
3	T7

STAFF

name	roomno	year
T2	IT206	NULL
T1	2.26	2
T3	2.82	NULL
T4	2.34	1
T5	IT212	NULL
T6	IT204	NULL
T7	A14	3
T8	2.10	NULL
T9	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



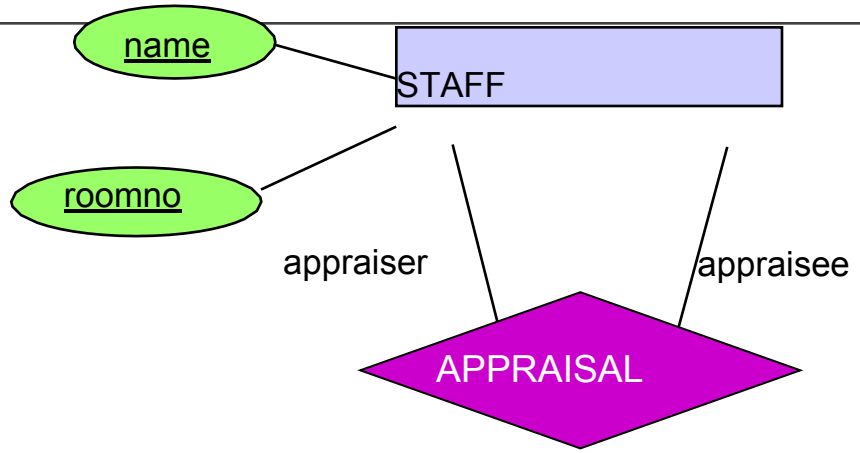
STUDENT

studno	given	family
s1 s2	prakash	tyagi
	mukesh	sharma

STUDENT_CONTACT

studno	contact
s1	Mr. tyagi
s1	Mrs tyagi
s2	Bill sharma
s2	Mrs tyagi
s2	Bilal-Ji

Mapping Roles and Recursive Relationships

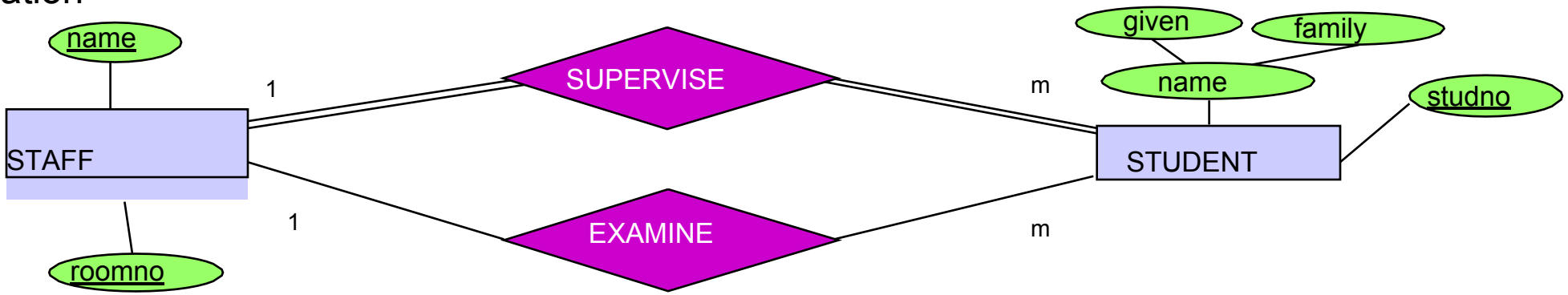


How can the entity STAFF *appear in both of its roles ?*

STAFF(name, roomno, appraiser, approomno)

Multiple Relationships between Entity Types

1. Treat each relationship type separately
2. Represent distinct relationships by different foreign keys drawing on the same relation



STAFF(name, roomno)

STUDENT(studno, given, family, ???)

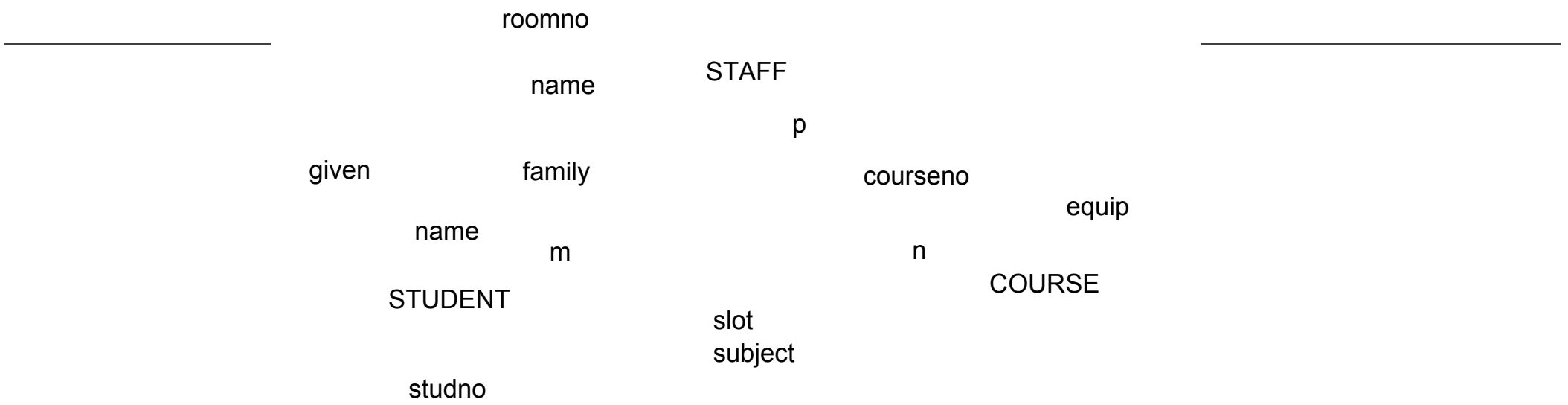
STUDENT(studno, given, family, ???)

EXAMINER(???)

SUPERVISOR(???)

EXAMINER-SUPERVISOR(???)

Non-binary Relationship



COURSE(<u>courseno</u>, subject, equip) STUDENT(<u>studno</u>, givenname, familyname) STAFF(<u>staffname</u>, <u>roomno</u>)

TUTORS(*???*

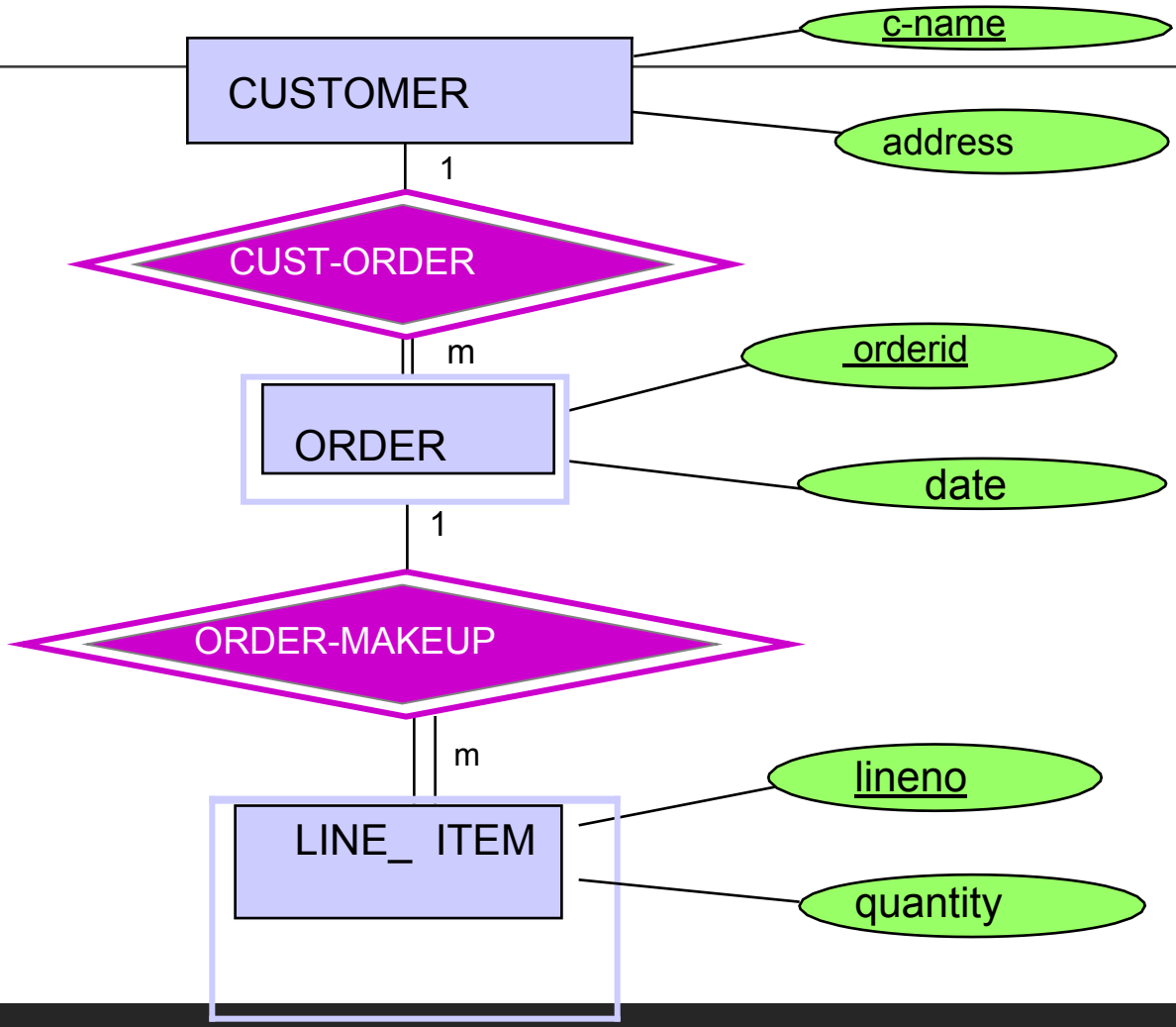
)

Weak Entities

- Strong entity type
- Identifying entity for ORDER
- Identifying entity for LINE_ITEM

- Weak entity type
- Identifying entity for LINE_ITEM

- Weak entity type



Mapping Weak Entities to Relations

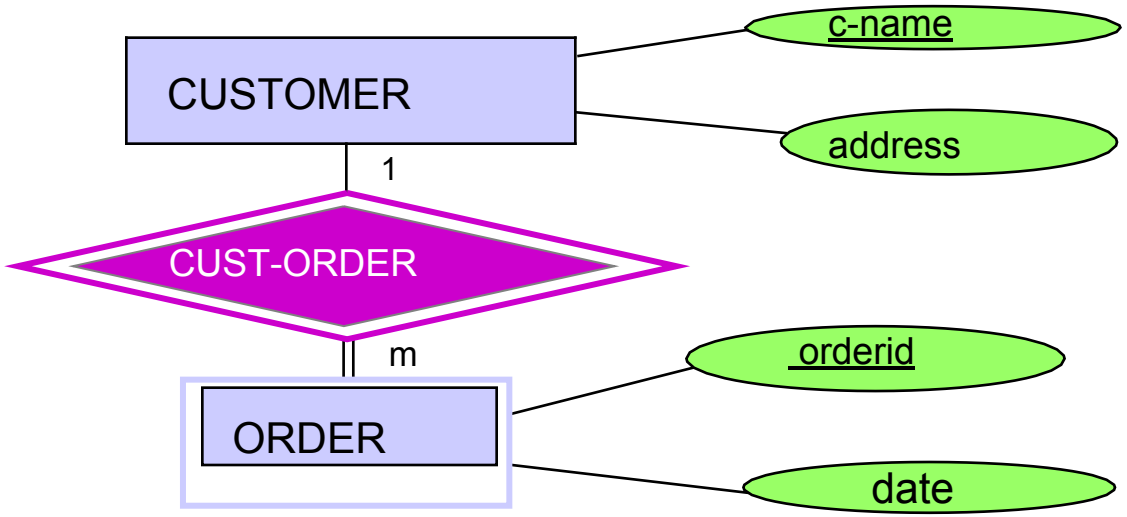
Create a relation with the attributes:

$$\text{primary\_key}(E_0) \cup \bigcup_{i=1}^{n} \text{discriminator}(E_i) \cup \{a_1,\ldots,a_n\}$$

primary key of identifying strong entity type

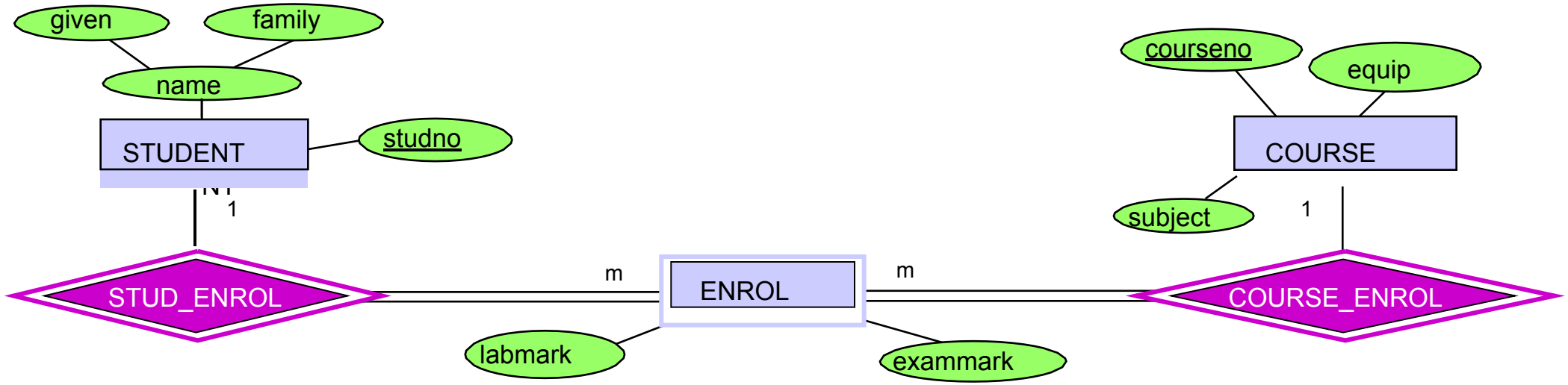
Discriminators of identifying weak entity types

Attributes of the weak entity type



Association Entity Types

An entity type that represents a relationship type:

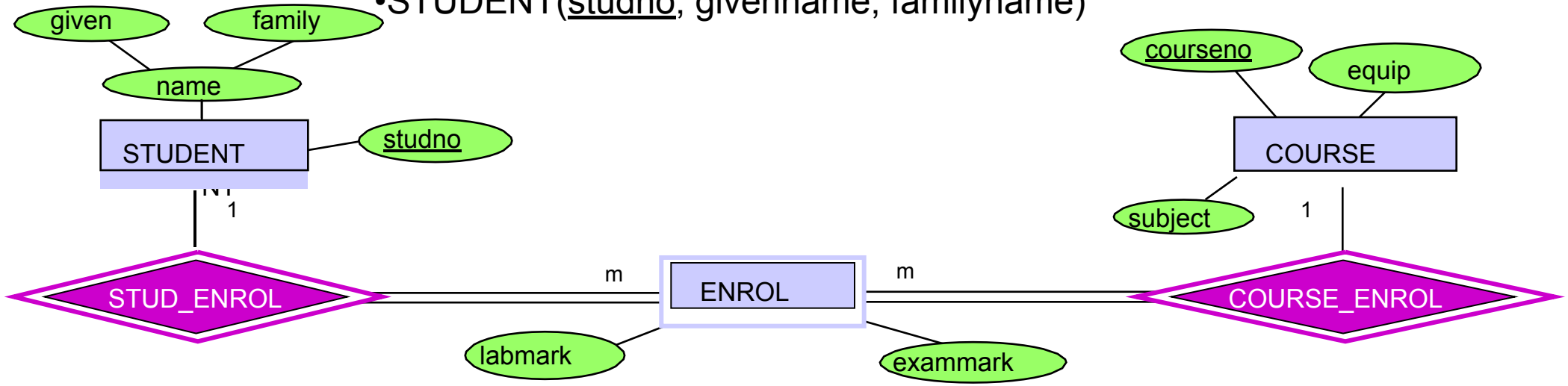


Association Entity Types

We have:

•COURSE(courseno, subject, equip)

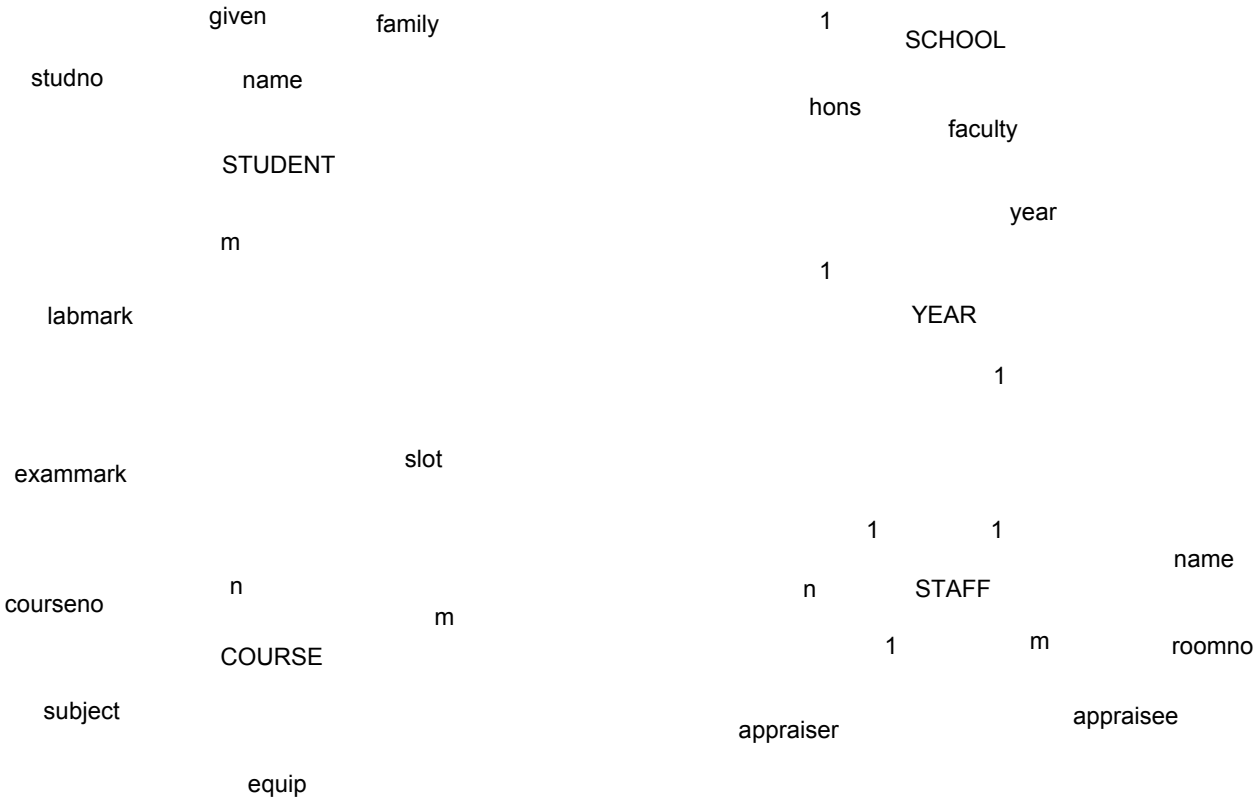
•STUDENT(studno, givenname, familyname)



Then:

•ENROL(courseno, studno, labmark, exammark)

Translation of the University Diagram



STUDENT
(<u>studno</u>, givenname, familyname, hons, tutor, tutorroom, slot, year)

ENROL(<u>studno, courseno</u>, labmark,exammark)

COURSE(<u>courseno</u>, subject, equip)

STAFF(<u>lecturer,roomno</u>, appraiser, approom)

TEACH(<u>courseno, lecturer,lecroom</u>)

YEAR(<u>year</u>, yeartutor, yeartutorroom)

SCHOOL(<u>hons</u>, faculty)

Exercise: Supervision of PhD Students

A database needs to be developed that keeps track of PhD students:

- For each student store the name and matriculation number. Matriculation numbers are unique.
- Each student has exactly one address. An address consists of street, town and post code, and is uniquely identified by this information.
- For each lecturer store the name, staff ID and office number. Staff ID's are unique.
- Each student has exactly one supervisor. A staff member may supervise a number of students.
- The date when supervision began also needs to be stored.

Exercise: Supervision of PhD Students

- For each research topic store the title and a short description. Titles are unique.
- Each student can be supervised in only one research topic, though topics that are currently not assigned also need to be stored in the database.

Tasks:

a) Design an entity relationship diagram that covers the requirements above. Do not forget to include cardinality and participation constraints.

b) Based on the ER-diagram from above, develop a relational database schema. List tables with their attributes. Identify keys and foreign keys.

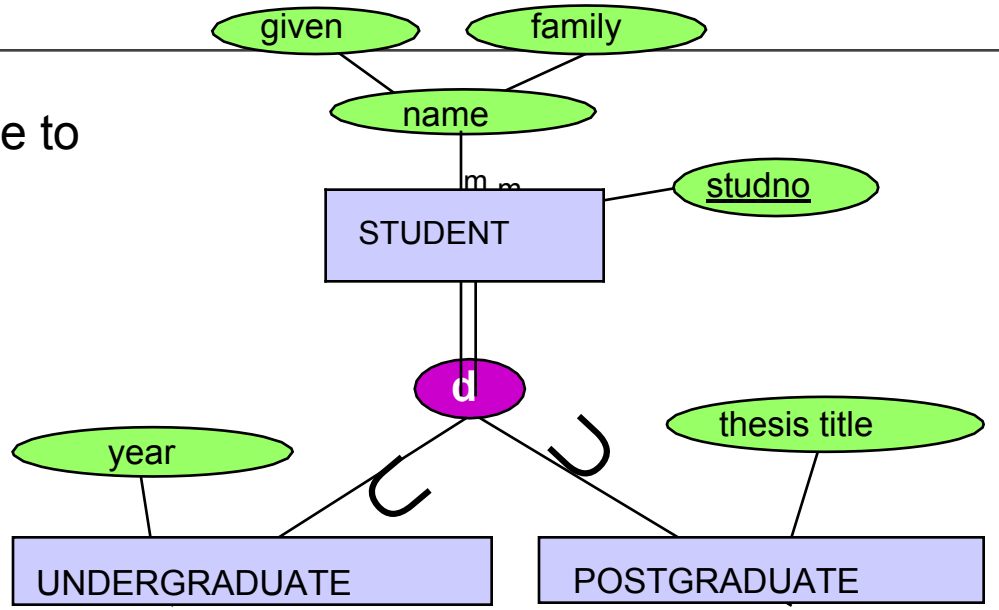
Translating of Hierarchies: Options

To store information about these classes, We have to define appropriate relations.

For each relation, we have to define:

- set of attributes
- primary key

In principle, there are three options:



A. Create a relation for each entity type in the schema, i.e., for both, superclass and subclasses
B. Create only relations for subclasses
C. Create only one relation, for the superclass

Translation into Relations: Option A

1. Create a relation for the superclass
2. For each subclass, create a relation over the set of attributes

primary_key(superclass) U attributes of subclass

The key for each subclass relation is: primary_key(superclass)

Inclusion constraint (foreign keys):

$$\pi_{key}(\text{superclass}) \supseteq \pi_{key}(\text{subclass}_i)$$

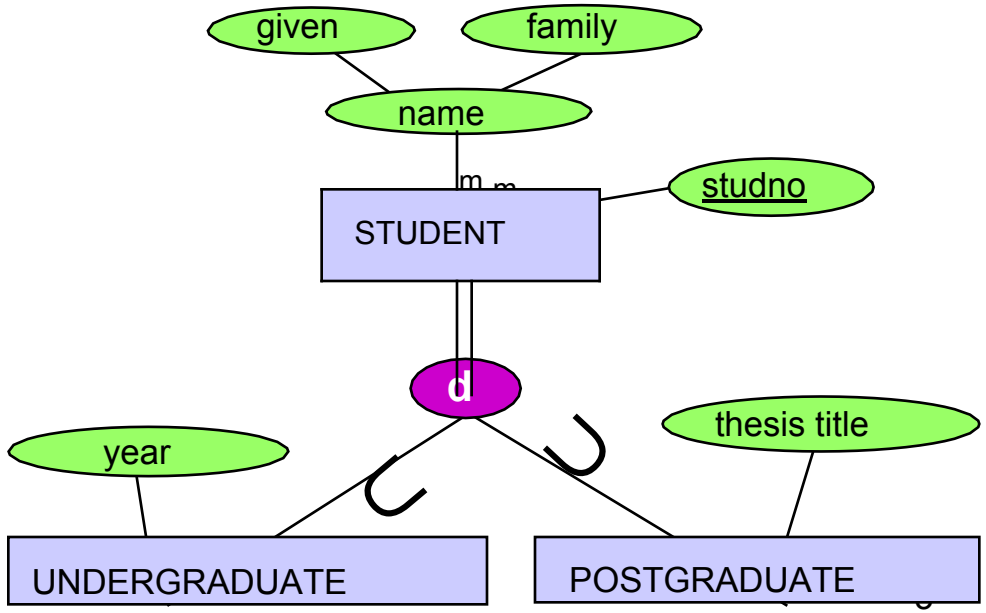
Covering constraint *(n = number of subclasses)*: n

$$\bigcup_{i=1} \pi_{key}(\text{subclass}_i) \supseteq \pi_{key}(\text{superclass})$$

Disjointness constraint:

$$\pi_{key}(\text{subclass}_i) \cap \pi_{key}(\text{subclass}_j) = \varnothing$$

for $i \neq j$



Translation into Relations: Option B

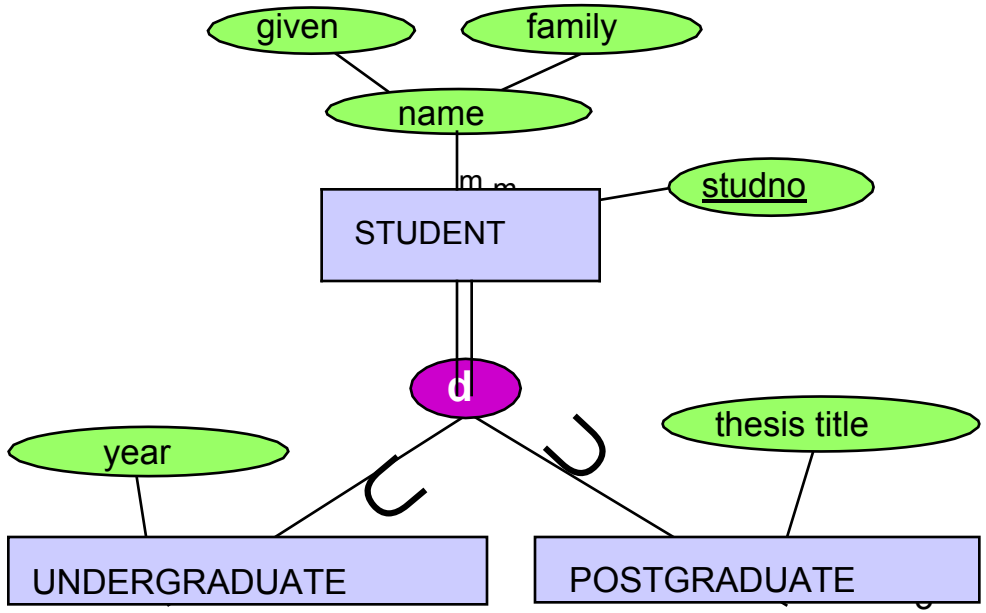
Create only relations for subclasses. Each subclass becomes a relation over the set of attributes:

attributes of superclass U attributes of subclass

The key for each subclass relation is: primary_key(superclass)

- Works only if coverage is total and disjoint
- *Partial coverage*: entities that are not in a subclass are lost
- *Overlapping classes*: redundancy
- Recovery of the superclass: OUTER UNION on the subclass relations

Codd: union that extends the schema to all common attributes



Translation into Relations: Option C

Create a single relation over the set of attributes

n

attributes of superclass U ⋃ attributes of subclass$_i$ i=1 U { *class* }

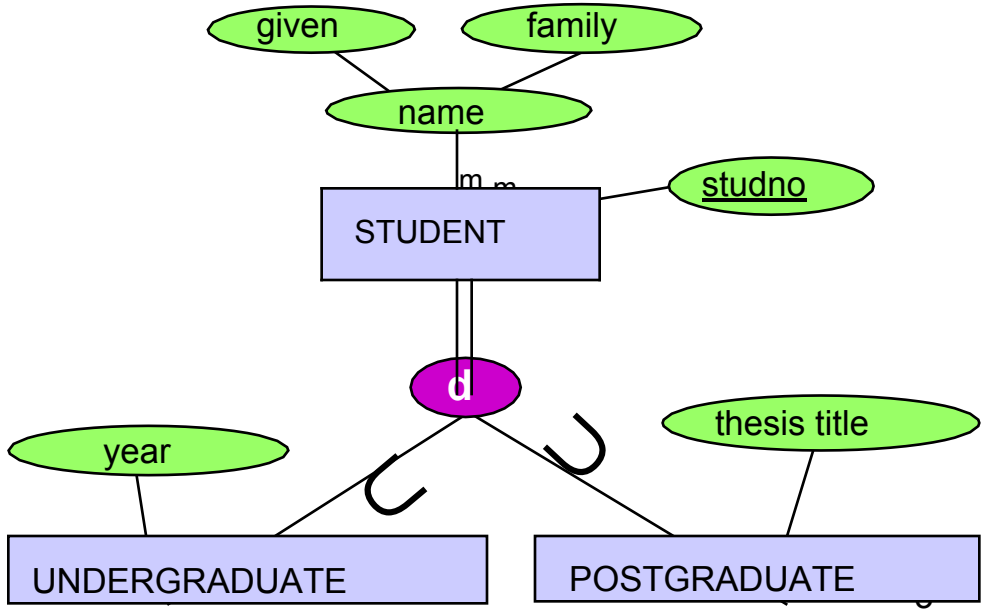
The key is: primary_key(superclass)

- Drawback: many 'not-applicable' nulls
- Benefit: No need for joins
- *Disjoint coverage*: one attribute *class* which indicates the subclass the tuple represents
- *Overlapping coverage: class*

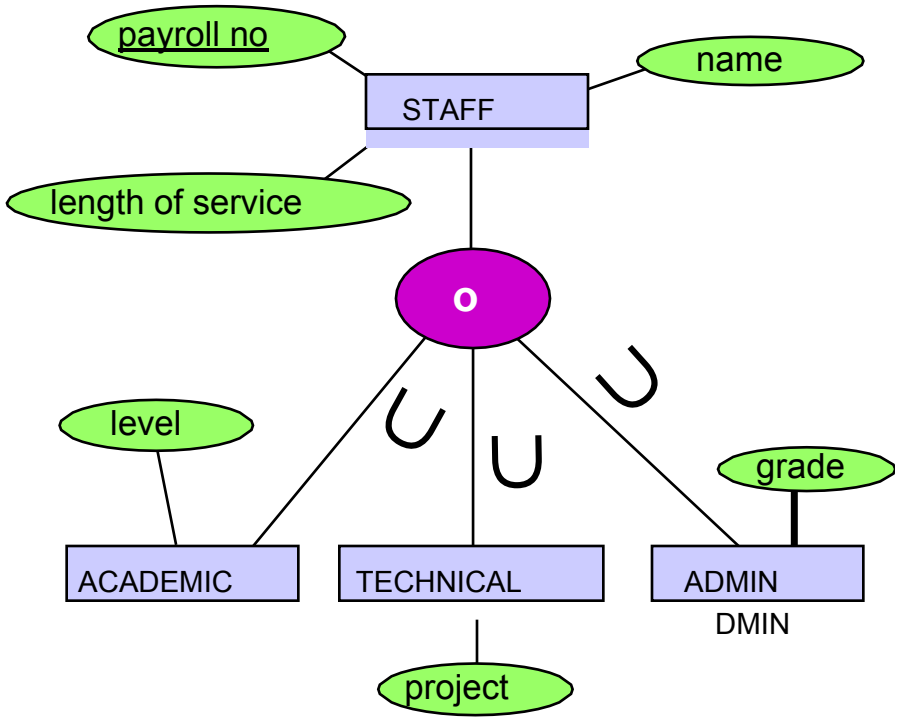
has to represent a set of classes

- *Partial coverage*: *class* is null

∴ entity is from superclass



Applying the Three Translations (Overlapping Coverage)



A. STAFF(payrollno, name, lengthOfService)
ACADEMIC(payrollno, level) TECHNICAL(payrollno, project) ADMIN(payrollno, grade)

B. ACADEMIC(payrollno, name, lengthOfService, level) TECHNICAL(payrollno, name, lengthOfService, project) ADMIN(payrollno, name, lengthOfService, grade)

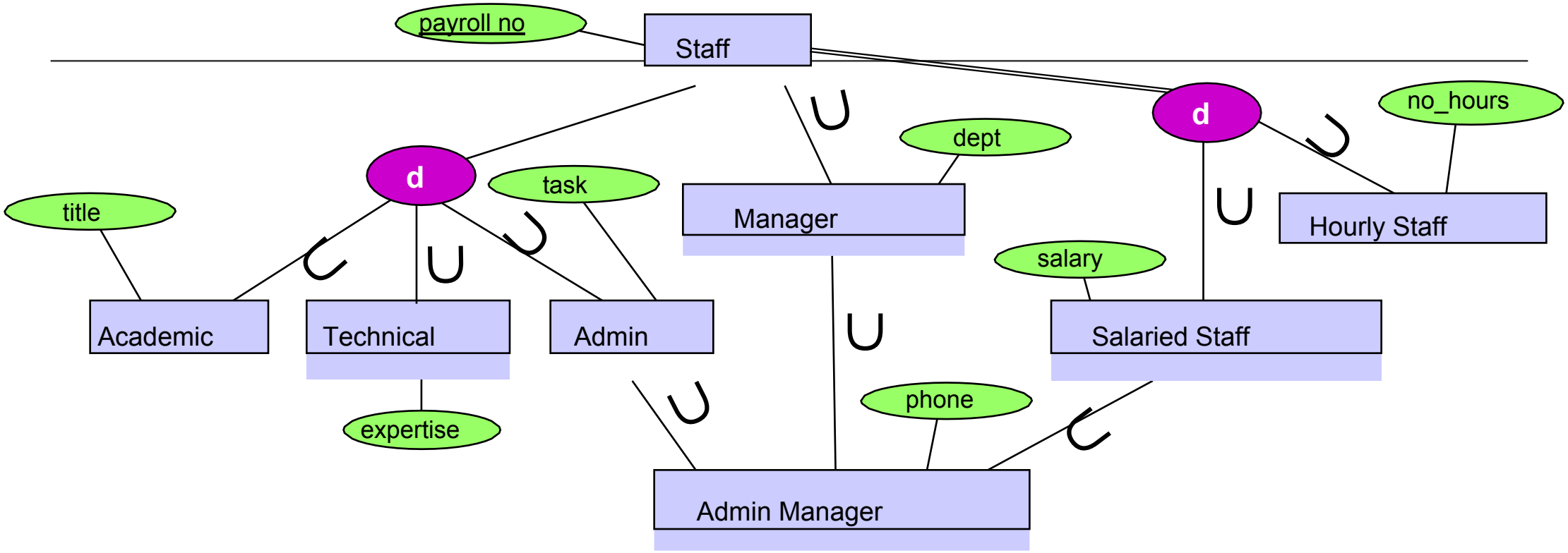
C. STAFF(payrollno, name, lengthOfService, level, project, grade, class1, class2, class3)

or

STAFF(payrollno, name, lengthOfService, level, project, grade, class)

class = powerset of classes

Specialisation Lattice with Shared Subclass



Exercise: For each of the approaches A, B, C, decide

- Which tables need to be created?
- Which are the attributes? And which are their possible values?

References

These Slides were prepared using following resources:

Books:

- A First Course in Database Systems, by J. Ullman and J. Widom
- Fundamentals of Database Systems, by R. Elmasri and S. Navathe