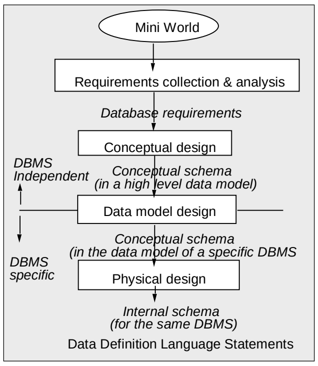
(Extended) Entity Relationship Modelling
and
Mappings to the Relational Data Model

Simplified phases of Database Design


Mini World
Requirements collection & analysis
Database requirements
Conceptual design
DBMS Independent
Conceptual schema (in a high level data model)
Data model design
DBMS specific
Conceptual schema (in the data model of a specific DBMS
Physical design
Internal schema (for the same DBMS)
Data Definition Language Statements

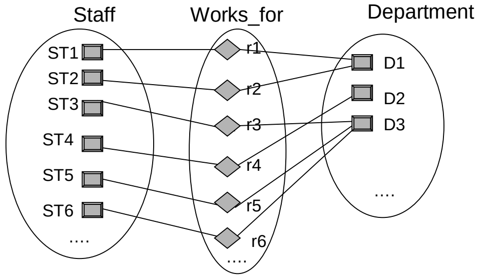
Conceptual Data Model Concepts

- "There exist things which have certain properties and which may be related in some way(s) to other things. Data represents specific facts about the things"
- Entity
  - thing or object that exists in its own right and is distinguishable, represented by an *Entity Type* of which there will be many *Entity Instances…… physical objects, events, activities, associations*
- Relationship
  - an association between several entities represented by a *Relationship Type* of which there will be many *Relationship Instances*

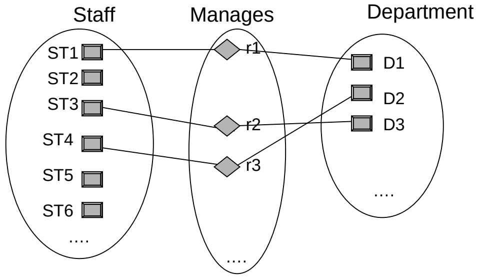
Conceptual Data Model Concepts

- Attribute
  - fact about an Entity Type or Relationship Type
  - an entity is often expressed as a set of attributes
- Entity Set or Extent
  - Set of all *Entity Instances* of the same *Entity Type*
- Relationship Set or Extent
  - Set of all *Relationship Instances* of the same *Relationship Type*

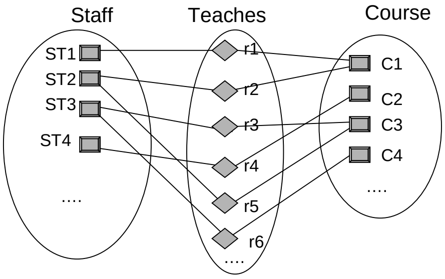
Entity Types and Relationship Types



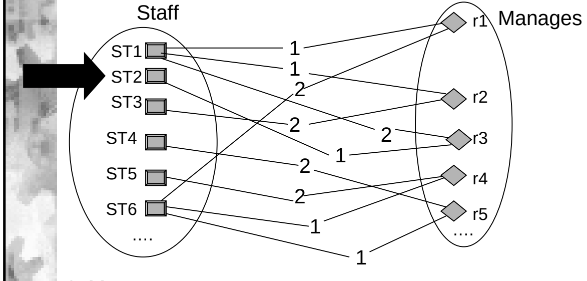
Optional Relationship Types



Many:many Relationship Types

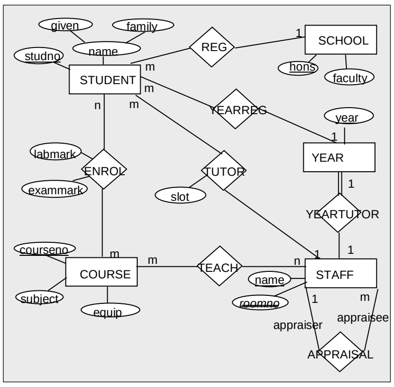


Recursive Relationship Types



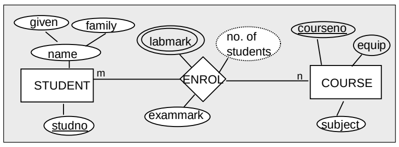
1. Manager
2. Employee

Entity Relationship Model



Attributes in Conceptual Modelling

- For each and every attribute must define *domain, data type, format* and whether it can be *null*
- Every entity type must have a *key* attribute or set of attributes
- *Composite or Atomic*
- *Single-valued or Multi-valued*
- *Derived*
- *Null valued*



Properties of Relationship Types

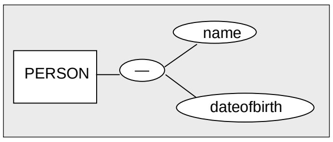
- Degree
  - The number of participating entity types
- Cardinality ratios
  - The number of instances of each of the participating entity types which can partake in a single instance of the relationship type *1:1, 1:many, many:1, many:many*
- Participation (optionality)
  - The relationship instance doesn't have to exist
  - Whether an entity instance has to participate in a relationship instance
- Role
  - The function that a particular entity type plays in a relationship type

Semantic Data Models

Extended-Entity-Relationship Modelling
Entity Attribute Relationship Modelling
Entity Relationship Attribute Modelling
Entity Modelling
Object Modelling
IFO,
NIAM etc.…
Extensions for temporal, constraints, rules etc

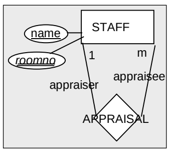
Chen 1976
Entity Relationship Modelling

Composite Keys



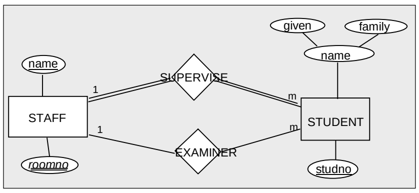
Roles & Recursive Relationships

- The function of an entity type in a relationship type

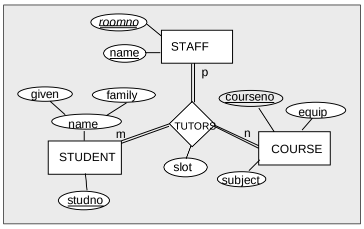


Roles & Association Relationships

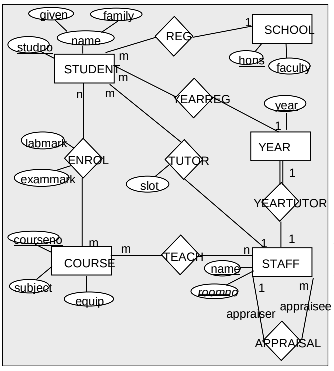
- The function of an entity type in a relationship type



Non-binary Relationship



Entity Relationship Model



STUDENT
(studno, givenname, familyname, hons, tutor, slot, year)

ENROL(studno, courseno, labmark,exammark)

COURSE(courseno, subject, equip)

STAFF(lecturer, *roomno*, appraiser)

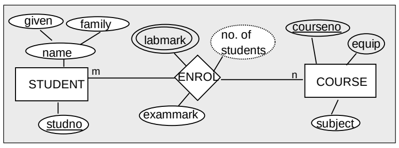
TEACH(courseno, lecturer)

YEAR(year, *yeartutor*)

SCHOOL(hons, faculty)

Mapping Entity Types to Relations

- For every entity type create a relation
  { primary_key (E) U {$a_1 \ldots a_m$} }
- Every attribute in entity becomes a relation attribute
- The relation is a subset of the X of the domains of the attributes
- Composite attributes—just include all the atomic attributes
- Derived attributes are not included but their derivation rules are



Mapping many:many Relationship Types to Relations

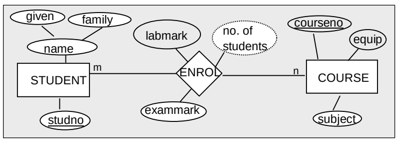
- Create a relation:

n (degree of relationship)

$$\bigcup_{i=1}^{n} \text{primary\_key}(E_i) \;\; \text{U} \;\; \{a_1 \ldots a_m\}$$

primary keys of each participating entity type in the relationship

attributes on the relationship type (if any)



Mapping one:many Relationship Types to Relations

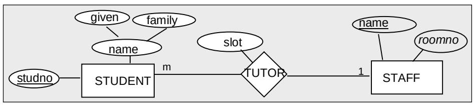
- Mostly: ‘*Posting the primary key*’
- Given E1 at ‘many’ end of relationship and E2 at ‘one’ end of relationship, add to the relation for E1
- Make the primary key of the entity at the ‘one’ end (the determined entity) a foreign key in the entity at the ‘many’ end (the determining entity). Include any relationship attributes with the foreign key entity

$$\{ \text{E1} \;\; \text{U} \;\; \text{primary\_key(E2)} \;\; \text{U} \;\; \{a_1 \ldots a_n\} \}$$

relation for entity E1

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

attributes on the relationship type (if any)



Mapping one:many Relationship Types to Relations

STUDENT				
studno	given	family	tutor	slot
s1	fred	jones	bush	12B
s2	mary	brown	kahn	12B
s3	sue	smith	goble	10A
s4	fred	bloggs	goble	11A
s5	peter	jones	zobel	13B
s6	jill	peters	kahn	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 one:many Relationship Types to Relations

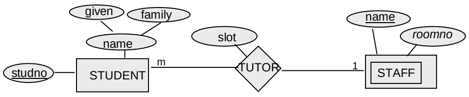
- Sometimes...
- If relationship type is optional to both entity types and an instance of the relationship is rare, and there are lots of attributes on the relationship then…
- Create a relation for the relationship type:

$$\{\text{primary\_key(E1)} \text{ U } \text{primary\_key(E2)} \text{ U } \{a_1 \ldots a_m\}$$

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 one:many Relationship Types to Relations

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

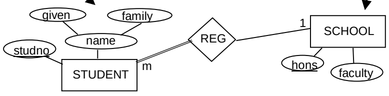
TUTOR		
studno	tutor	slot
s1	bush	12B
s2	kahn	12B
s3	goble	10A
s4	goble	11A
s5	zobel	13B
s6	kahn	12A

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

Optional Participation of Determined Entity (‘one end’)

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

A school entity instance does not have to participate in a relationship instance of REG



- SCHOOL(hons,faculty)
- STUDENT(studno,givenname,familyname, ???)

Optional Participation of Determined Entity

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

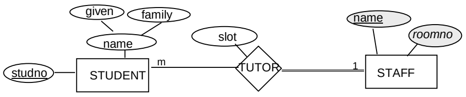
hons can't be *null* because it is mandatory for a student to be registered for a school.

SCHOOL

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

no-one registered for mi so doesn't occur as a foreign key value

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



A student entity instance does not have to 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')

1. STUDENT (studno,givenname,familyname,tutor,slot)
   STAFF(name, roomno)
   Integrity constraints:
   $\pi_{(name)}$ STAFF – $\pi_{(tutor)}$ STUDENT = ∅

2. STUDENT(studno,givenname,familyname)
   STAFF(name,roomno)
   TUTOR(studno,tutor,slot)

3. same as 2 if lots of attributes on TUTOR

Optional Participation of the Determinant Entity

STUDENT

studno	given	family	tutor	slot
s1	fred	jones	bush	12B
s2	mary	brown	kahn	12B
s3	sue	smith	goble	10A
s4	fred	bloggs	*null*	*null*
s5	peter	jones	zobel	13B
s6	jill	peters	*null*	*null*

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 one:one Relationship Types to Relations

1. 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*

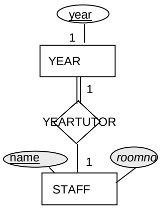
2. Merge the entity types together

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

YEAR

year	*yeartutor*
1	zobel
2	bush
3	capon



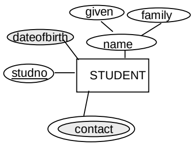
Multi-Valued Attributes

- Create a relation for each multi-valued attribute { primary_key(E_i) U multi-valued attribute }

The primary key is (primary_key(E_i) U multi-valued attribute)

STUDENT

studno	given	family	dateofbirth	contact
s1	fred	jones	10/4/78	Mr. Jones Mrs Jones
s2	mary	brown	12/1/72	Bill Brown Mrs Jones Billy-Jo Woods

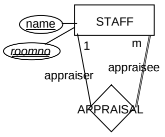


STUDENT_CONTACTS

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

Mapping Roles & Recursive Relationships

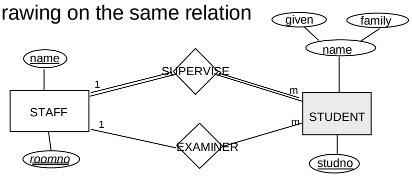
- The function of an entity type in a relationship type



STAFF(name, roomno, ???)

Multiple Roles between Entity Types

1. Treat each relationship type separately
2. Distinct roles are represented by different foreign keys drawing on the same relation

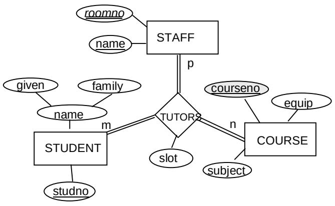


STAFF(name,roomno)
STUDENT(studno,given,family, ???)

STAFF(name,roomno)
EXAMINER(???)
SUPERVISOR(???)

EXAM-SUPER(???)

Non-binary Relationship



COURSE(courseno, subject, equip)
STUDENT(studno, givenname, familyname)
STAFF(staffname, roomno)
TUTORS(???)

Comparative Terms

EER	Relational	
	Formal	Informal
Entity Type Schema	Relational Schema	Table description
Entity Type	Relation	Table
Entity instance	Tuple	Row
1:many relationship type	Use foreign keys	Use foreign keys
1:many relationship instance	Use foreign keys	Use foreign keys
Attribute	Attribute	Column
Domain or Value Set	Domain	Data Type
Key	Candidate Key	Candidate Key
No equivalent	Primary Key	Primary Key
Multivalued attribute	No equivalent	No equivalent
Composite attribute	No equivalent	No equivalent

Superclasses, Subclasses; Specialisation & Generalisation Relationships

- Subclasses and Superclasses
  - a subclass entity type is a specialised type of superclass entity type
  - a subclass entity type represents a subset or subgrouping of superclass entity type's instances
  - e.g. undergraduates and postgraduates are subclasses of student superclass
- Attribute Inheritance
  - subclasses inherit properties (attributes) of their superclasses

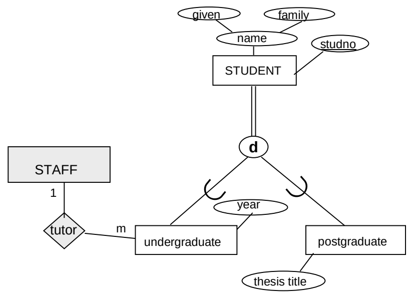
Constraints on Specialisation & Generalisation

- Specialisation
  - the process of defining a set of more specialised entity types of an entity type
- Generalisation
  - the process of defining a generalised entity type from a set of entity types
- Predicate/Condition defined
  - determine the entities that will become members of each subclass by a condition on an attribute value. All member instances of the subclass must satisfy the predicate
  - e.g. first years and second years are subclasses of undergraduates based on their year attribute.
- User defined
  - no condition for determining subclass membership

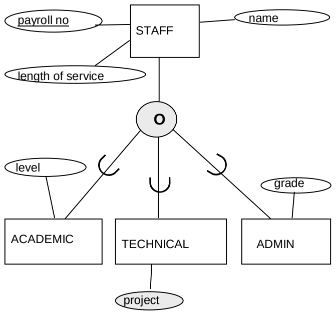
Constraints on Specialisation & Generalisation

- Disjointness
  - *Overlap*
    - the same entity instance may be a member of *more than one* subclass of the specialisation
  - *Disjoint*
    - the same entity instance may be a member of *only one* subclass of the specialisation
- Completeness
  - *Total*
    - every entity instance in the superclass *must be* a member of some subclass in the specialisation
  - *Partial*
    - an entity instance in the superclass need not be a member of any subclass in the specialisation

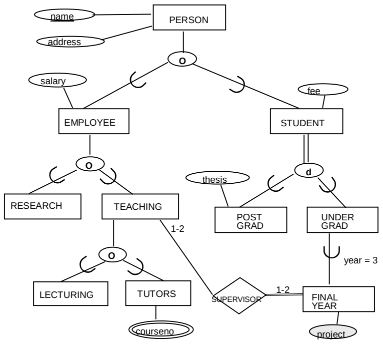
Specialisation & Generalisation Relationships



Superclasses, Subclasses
Specialisation & Generalisation Relationships

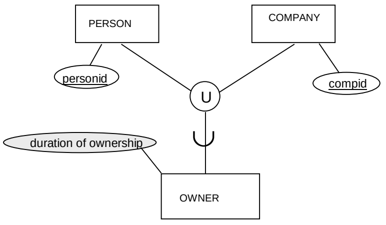


Superclasses, Subclasses
Specialisation & Generalisation Relationships



Categories and Categorisation

- a single superclass/subclass relationship with more than one superclass, where the superclasses represent different entity types (sometimes with different keys)



Specialisation & Generalisation Option A

1. Create a relation for superclass
2. Create a relation for each subclass such that:

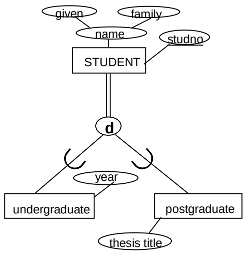
{primary_key of superclass} U {attributes of subclass}

key for subclass is (primary_key of superclass)

Inclusion dependency:
$\pi_{<key>}$(superclass) $\supseteq$ $\pi_{<key>}$(subclass)

Covering dependency:
n (number of subclasses)
$\bigcup$ $\pi_{<key>}$(subclass) = $\pi_{<key>}$(superclass)
i=1

Disjoint dependency:
n (number of subclasses)
$\bigcap$ $\pi_{<key>}$(subclass) = $\varnothing$
i=1

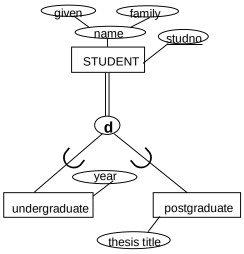


Specialisation & Generalisation Option B

1. Create a relation for each subclass such that:
{primary_key U {attributes U {attributes of
of superclass} of superclass} subclass}

key for each relation is (primary_key of superclass)

- Works for total and disjoint constraints
- *Partial*: lose any entity that is not in a subclass
- *Overlapping*: redundancy
- To recover the superclass can do an OUTER UNION on the subclass relations

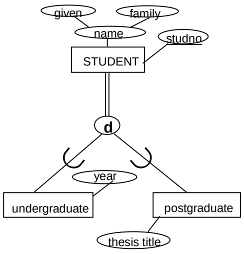


Specialisation & Generalisation Option C

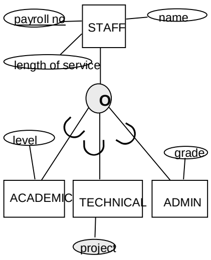
1. Create one relation such that:
{primary_key U {attributes U {attributes U {type
of superclass} of superclass} of all subclasses} attribute}

- key for subclass is (primary_key of superclass)

- Many ‘not-applicable’ nulls
- Does away with joins
- *Disjoint*: one type which indicates which subclass the tuple represents
- *Overlap:* set of types = number of subclasses
- *Partial*: type is null ∴ represents superclass



Specialisation & Generalisation Overlapping

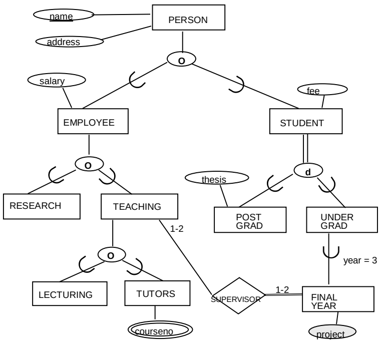


1.STAFF(payrollno,name,lengthofservice)
ACADEMIC(payrollno,level)
TECHNICAL(payrollno,project)
ADMIN(payrollno,grade)

2.ACADEMIC(payrollno,name,lengthofservice, level)
TECHNICAL(payrollno,name,lengthofservice, project)
ADMIN(payrollno,name,lengthofservice,grade)

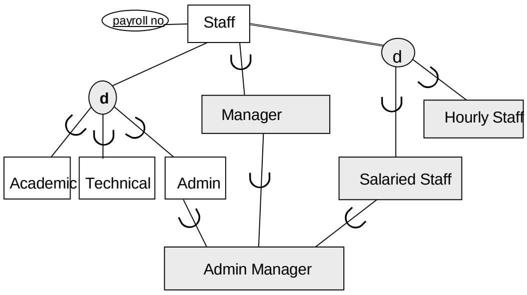
3.STAFF(payrollno,name,lengthofservice,level, project,grade,type1,type2,type3)
STAFF(payrollno,name,lengthofservice,level, project,grade,type)
type = powerset of classes

Specialisation & Generalisation Relationships



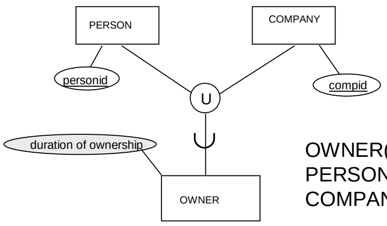
Specialisation Lattice with Shared Subclass

- To be a shared subclass the superclasses must have the same key, so any of the options A, B or C stand.



Categories and Categorisation

- A category is a subclass of the union of two or more superclasses that can have different keys because they can be of different entity types
- If defining superclasses have different keys, specify a new *surrogate* key



OWNER(???)
PERSON(???)
COMPANY(???)

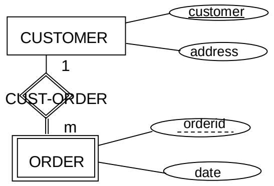
Entity Constraints

- If an entity instance X depends on the existence of an entity instance Y, then X is existence dependent on
- entity type Y is dominant
- entity type X is subordinate

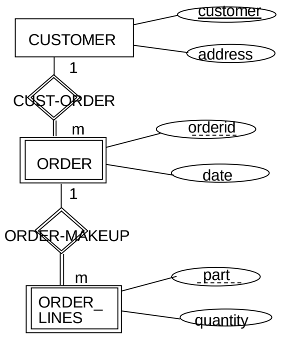
customer	orderid	date	part	quantity
RipOff Inc	123	23/6/94	widget thingie	20 24
RipOff Inc	678	3/10/94	widget	20
UpYa Ltd	123	27/9/94	wotsits widget thingie	800 50 24

Strong and Weak Entities (identifier dependency)

- a *strong* entity type has an identifying primary key
- a *weak* entity type does not have a primary key but does have a discriminator



Weak Entity



Mapping Weak Entities to Relations

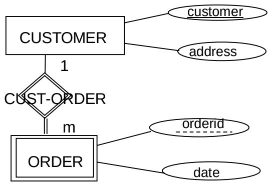
- Create a relation

$\bigcup_{i=1}^{n}$ primary_key(Ei) U partial_key U {ai…an}

Primary key of each participating identifying entity type

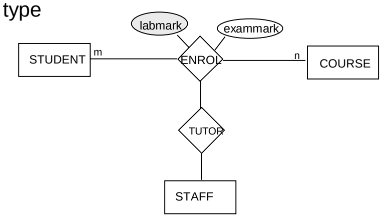
Partial key of weak entity (if any)

Attributes of the weak entity type (if any)



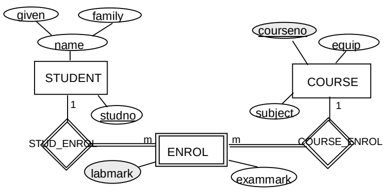
Association Entity Type

- An entity type that represents an association relationship type
- Useful if:
  - a relationship has lots of attributes
  - you want a relationship type with a relationship type



Association Entity Type plus Mapping

- An entity type that represents an association relationship type



COURSE(courseno, subject, equip)
STUDENT(studno, givenname, familyname)

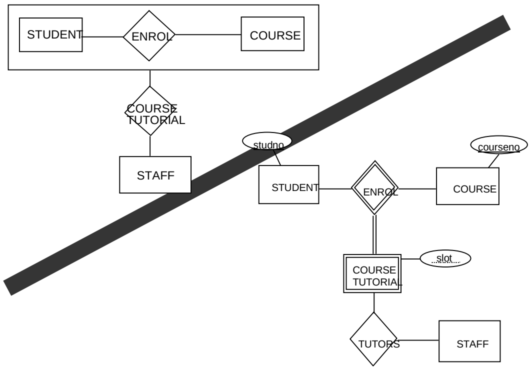
Aggregation

- Aggregation is an abstraction concept for building composite entities from their components

1. aggregate attribute values to form a whole entity
2. combining entities that are related by an association relationship into *higher-level aggregate entity*

- IS-A-PART-OF
- IS-A-COMPONENT-OF
- Sadly, not catered for in EER modelling.

Aggregation



Hints for EER Modelling

- identify entity types by searching for nouns and noun phrases
- assume all entities are strong and check for weak ones on a later pass
- need an identifier for each strong entity
- assume all relationships are partial participation (optional) and check for total (mandatory) ones on a later pass
- expect to keep changing your mind about whether things are entities, relationships or attributes
- keep level of detail relevant and consistent (for example leave out attributes at first)
- approach diagram through different views and merge them

Lets Practice!

- A record company wishes to use a computer database to help with its operations regarding its performers, recordings and song catalogue.
- Songs have a unique song number, a non-unique title and a composition date. A song can be written by a number of composers; the composer's full name is required. Songs are recorded by recording artists (bands or solo performers). A song is recorded as a track of a CD. A CD has many songs on it, called tracks. CDs have a unique record catalogue number, a title and must have a producer (the full name of the producer is required). Each track must have the recording date and the track number of the CD.
- A song can appear on many (or no) CDs, and be recorded by many different recording artists. The same recording artist might re-record the same song on different CDs. A CD must have only 1 recording artist appearing on it. CDs can be released a number of times, and each time the release date and associated number of sales is required.