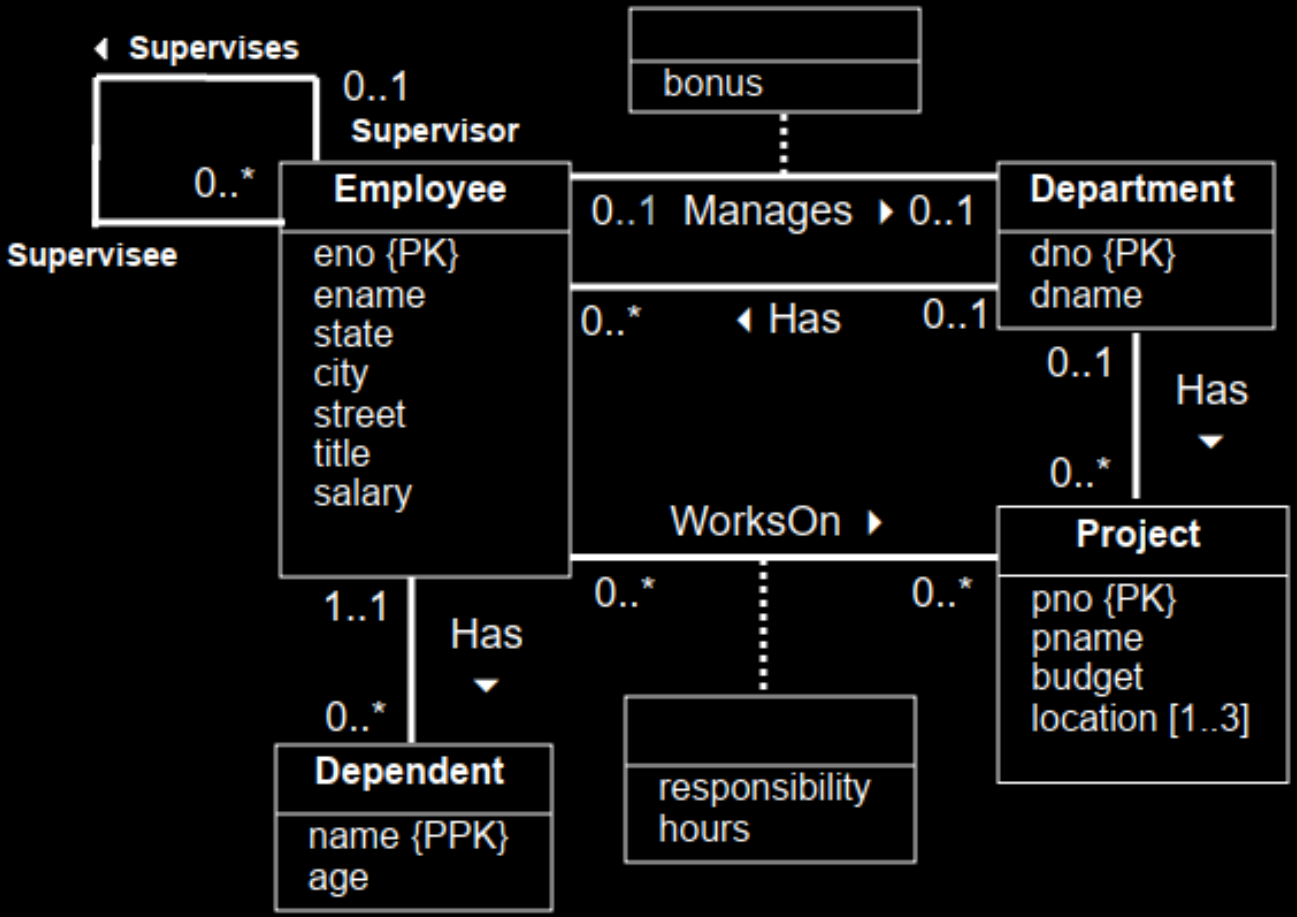
Lecture 10

ER Model to Relational Schemas

ER Model - Example

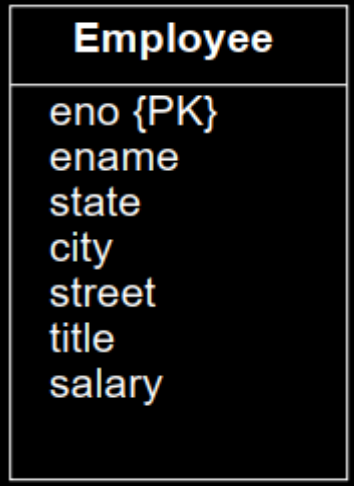


Steps

Entities

1. Convert **Strong Entities**
2. Convert **Weak Entities**
3. Convert **Multi - Valued Attributes**

Step#1 Convert each strong entity to a relation

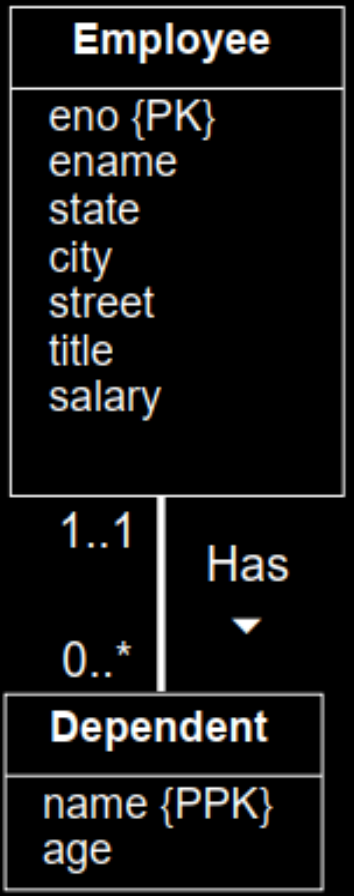


Employee (eno, ename, state, city, street, title, salary)

- Notes:
  - 1) Attributes of the entity type become attributes of the relation.
  - 2) Include only simple attributes in relation. For composite attributes, only create attributes in the relation for their simple components.
  - 3) Multi-valued attributes are handled separately
  - 4) The primary key of the relation is the key attributes for the entity.

Step#2

Convert each weak entity into a relation with foreign keys to its identifying relations (entities).



Employee (<u>eno</u>, ename, state, city, street, title, salary)

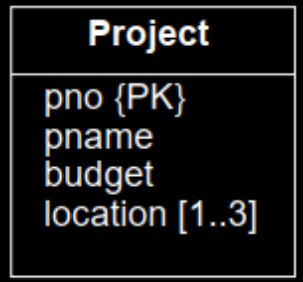
For each weak entity *W* with identifying owners E_1, E_2, …, E_n create a relation *R*:

- Identify relations R_1, R_2, …, R_n for entity types E_1, E_2, …, E_n.
- The primary key of *R* consists of the primary keys of R_1, R_2, …, R_n plus the partial key of the weak entity.
- Create a foreign key in *R* to the primary key of each relation R_1, R_2, …, R_n.
- Attributes are converted the same as strong entities.

Dependent (<u>eno</u>, <u>name</u>, age)

Step#3

Convert a multi-valued attribute into a relation with composite primary key consisting of the attribute value plus the primary key of the attribute's entity.



Project (<u>pno</u>, pname, budget, dno)

ProjectLocation (<u>pno</u>, <u>location</u>)

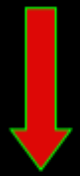
Project (<u>pno</u>, pname, budget, dno)

Step#3

Convert a multi-valued attribute into a relation with composite primary key consisting of the attribute value plus the primary key of the attribute's entity.

Project
pno {PK} pname budget location [1..3]

Project (<u>pno</u>, pname, budget, dno)



ProjectLocation (<u>pno</u>, <u>location</u>)

Project (<u>pno</u>, pname, budget, dno)

Given a multi-valued attribute *A* of entity E_i:

- Identify the corresponding relation R_i.
- Create a new relation *R* representing the attribute where:
  - *R* contains the simple, single-valued attribute *A*.
  - Add the primary key attributes of R_i to *R*, and create a foreign key reference to R_i from *R*.
  - The primary key of *R* is a composite key consisting of the primary key of R_i and *A*.

Steps - Continued

Relationships

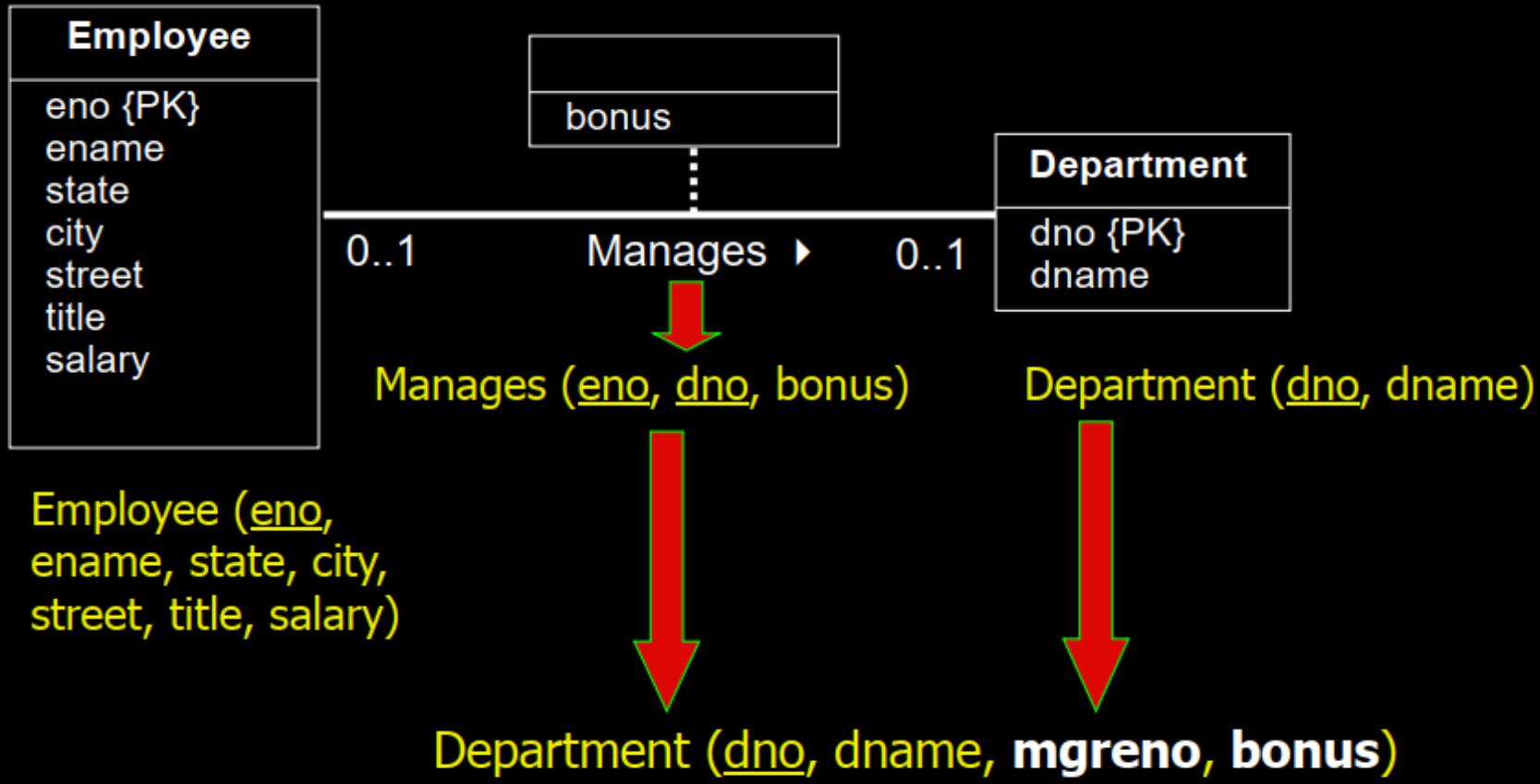
- Step 4. **One to One** relationships
- Step 5. **One to Many** relationships
- Step 6. **Many to Many** relationships
- Step 7. **n-ary** Relationships

Step#4

Convert binary 1:1 relationships into a UNIQUE foreign key reference from one relation to the other.

Given a binary 1:1 relationship R between two entities E_i and E_j:

- Identify the corresponding relations R_i and R_j.
- Chose one of the relations, say R_i, and:
  - Add the attributes of R to R_i.
  - Add the primary key attributes of R_j to R_i, and create a foreign key reference to R_j from R_i.
  - Declare these primary key attributes of R_j to be UNIQUE.



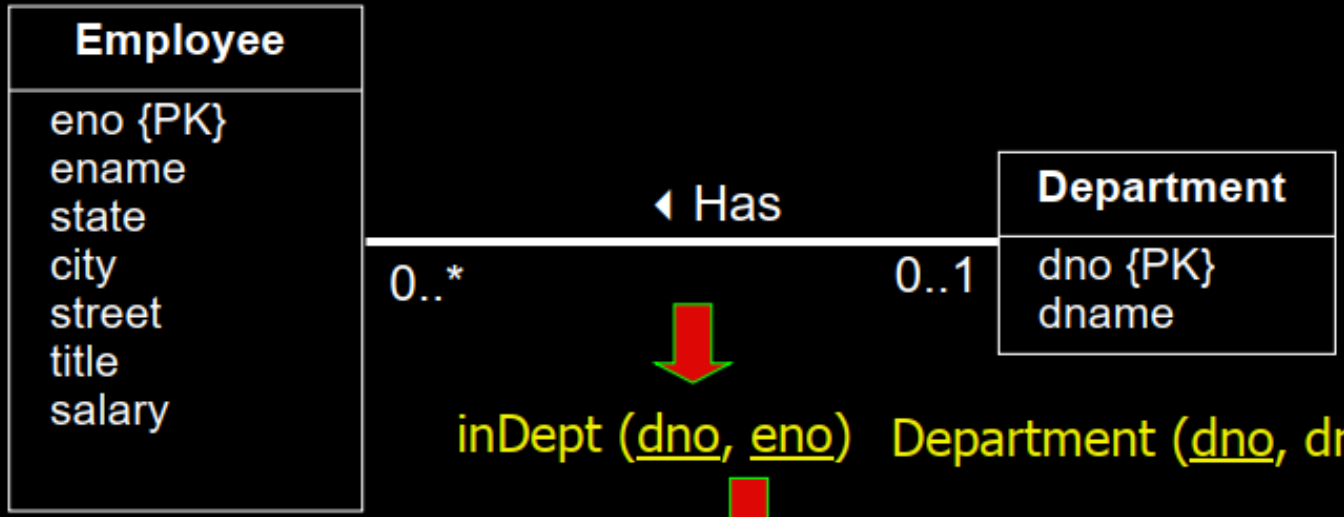
Note: Renamed `eno` to `mgreno` for clarity.

Step#5

Convert binary 1:N relationships between into a foreign key reference from the N-side relation to the 1-side relation.

Given a binary 1:N relationship R between two entities E_i and E_j:

- Identify the corresponding relations R_i and R_j.
- Let R_i be the N-side of the relation.
  - ⇨ Add the attributes of R to R_i.
  - ⇨ Add the primary key attributes of R_j to R_i, and create a foreign key reference to R_j from R_i.

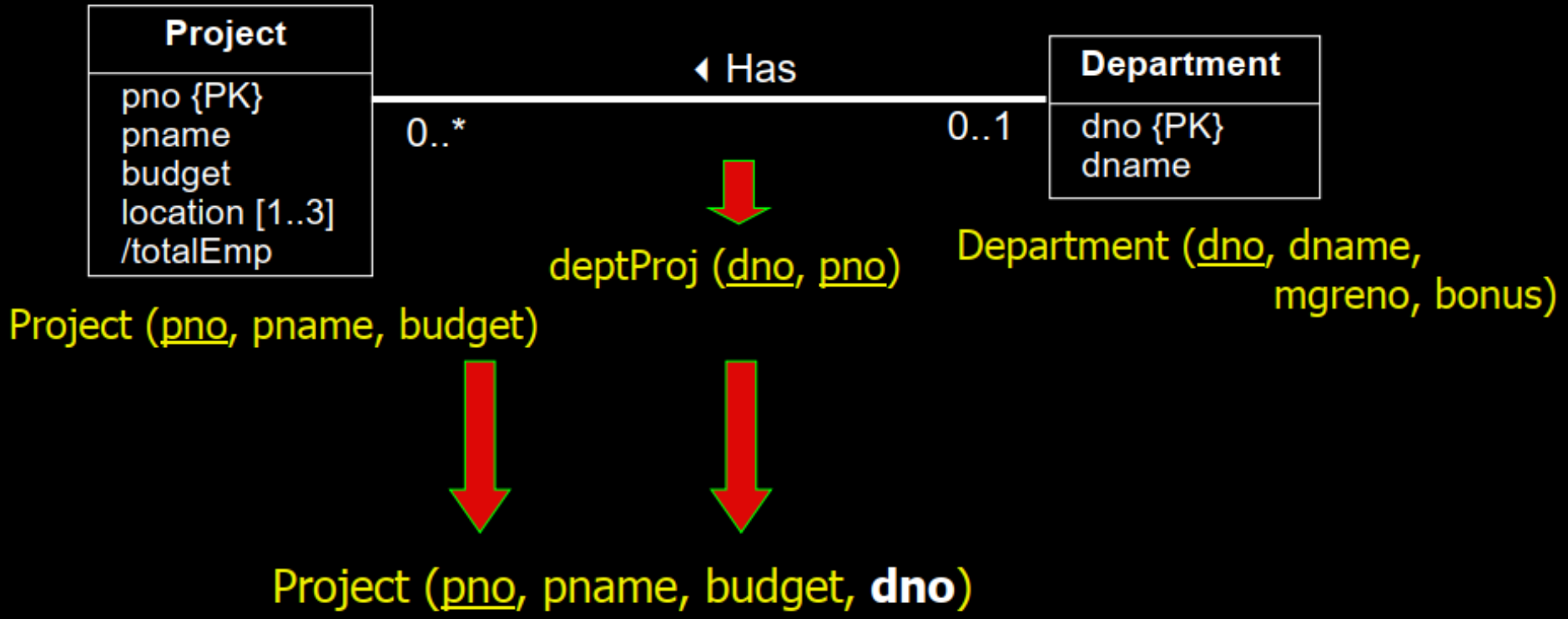


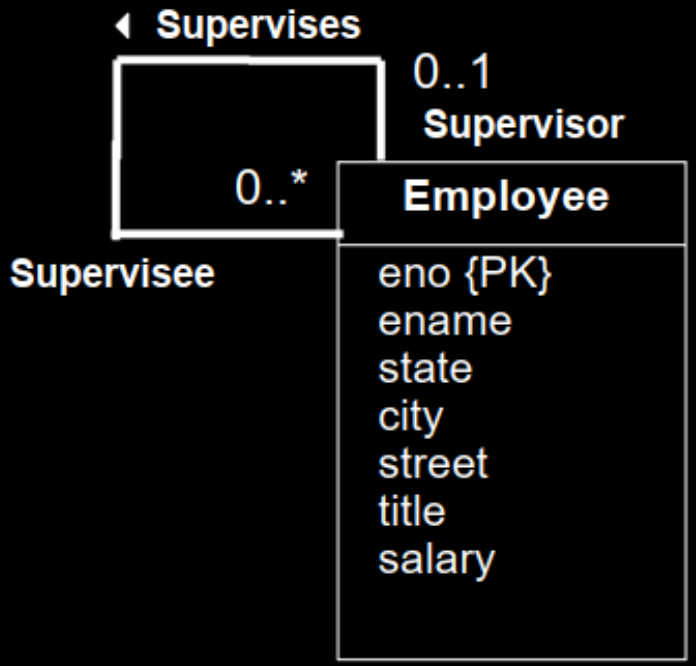
inDept (<u>dno</u>, <u>eno</u>)

Department (<u>dno</u>, dname, mgreno, bonus)

Employee (<u>eno</u>, ename, state, city, street, title, salary)

Employee (<u>eno</u>, ename, state, city, street, title, salary, **dno**)


Project
pno {PK}
pname
budget
location [1..3]
/totalEmp
Has
0..*
0..1
Department
dno {PK}
dname
deptProj (dno, pno)
Department (dno, dname, mgreno, bonus)
Project (pno, pname, budget)
Project (pno, pname, budget, dno)



Supervises (<u>supereno</u>, <u>eno</u>)

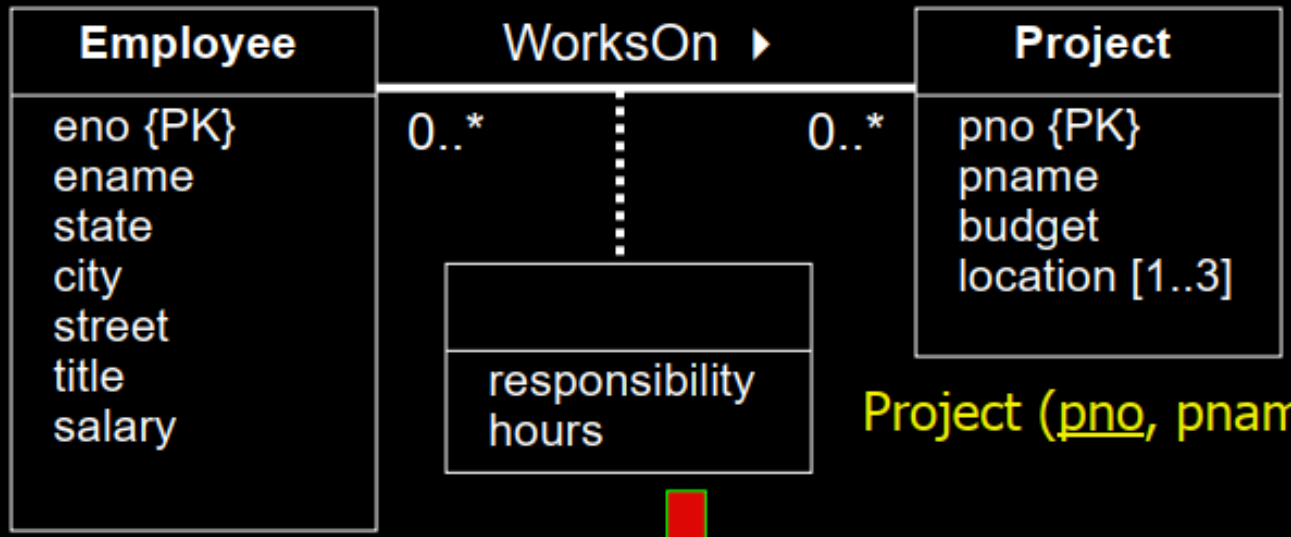
Employee (<u>eno</u>, ename, state, city, street, title, salary, dno)

Employee (<u>eno</u>, ename, state, city, street, title, salary, dno, **supereno**)

Step#6 Convert binary M:N relationships into a new relation with foreign keys to the two participating entities.

Given a binary M:N relationship between entities E_i and E_j:

- Identify the corresponding relations R_i and R_j.
- Create a new relation R representing the relationship where:
  - R contains the relationship attributes.
  - The primary key of R is a composite key consisting of the primary keys of R_i and R_j.
  - Add the primary key attributes of R_i and R_j to R, and create a foreign key reference to R_i from R and to R_j from R.

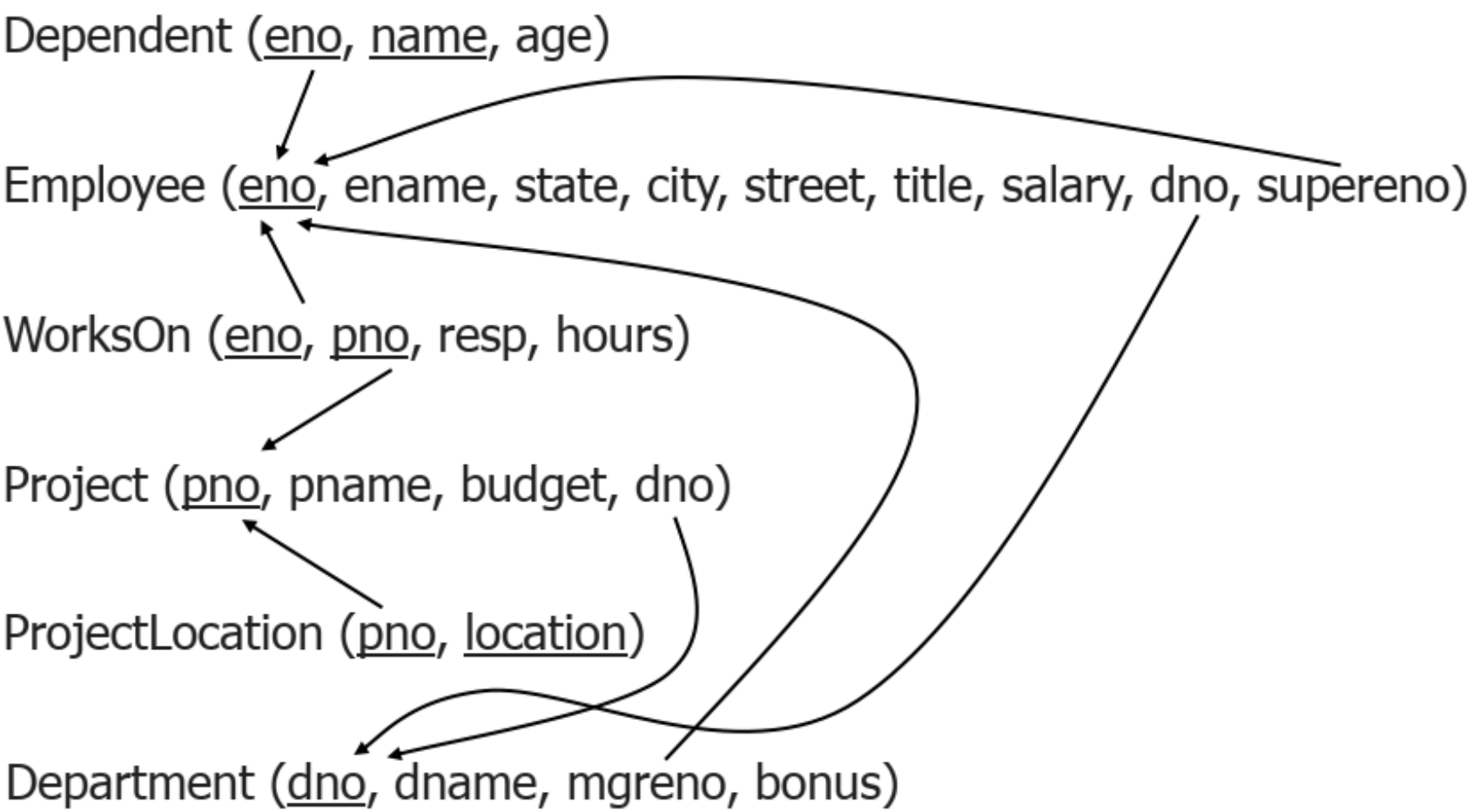


Project (pno, pname, budget, dno)

Employee (eno, ename, state, city, street, title, salary, dno, supereno)

WorksOn (eno, pno, resp, hours)

Relational Schema

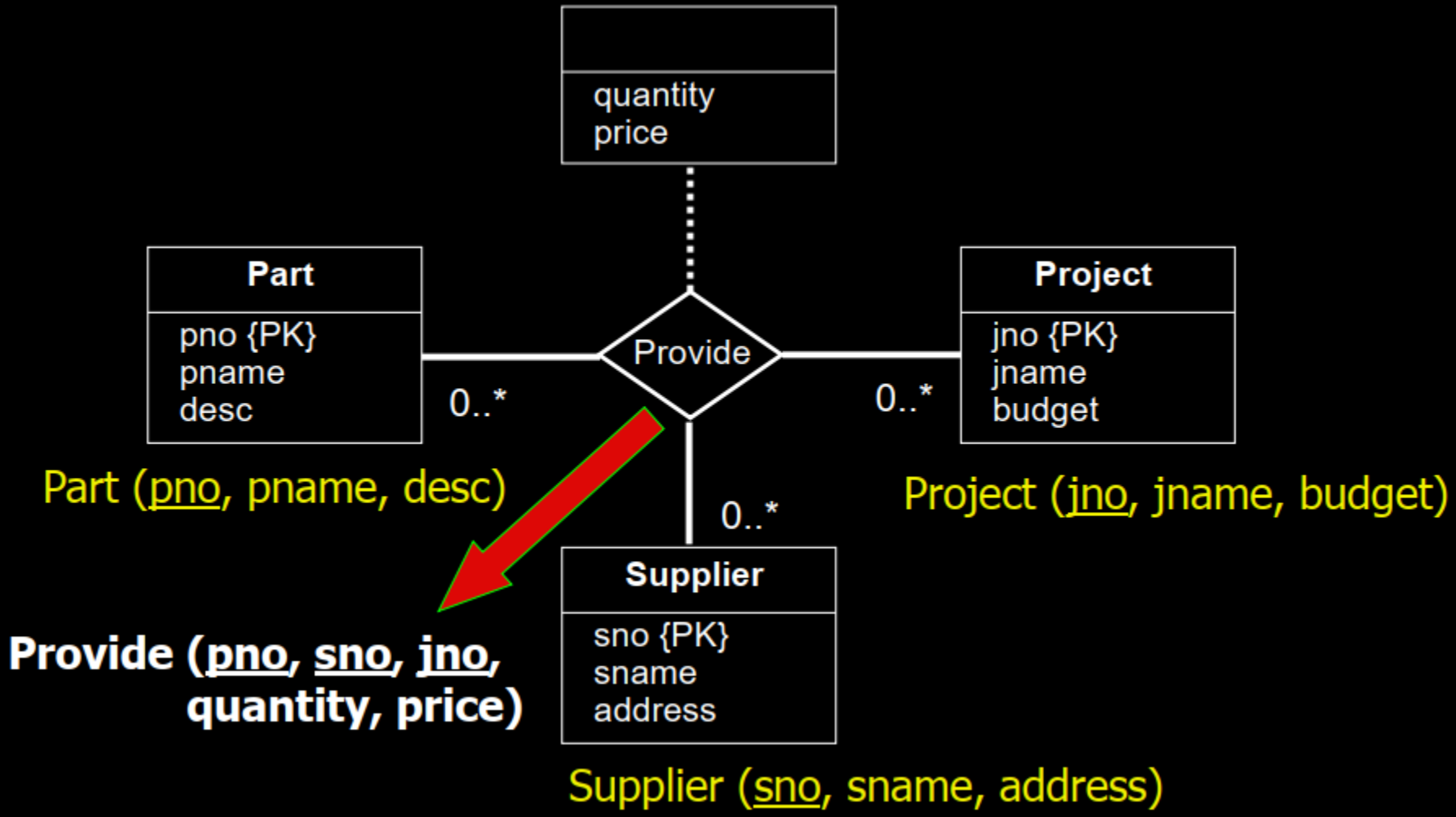


Step#7

Convert *n*-ary relationships by creating a new relation to represent the relationship and creating foreign keys that reference the related entities.

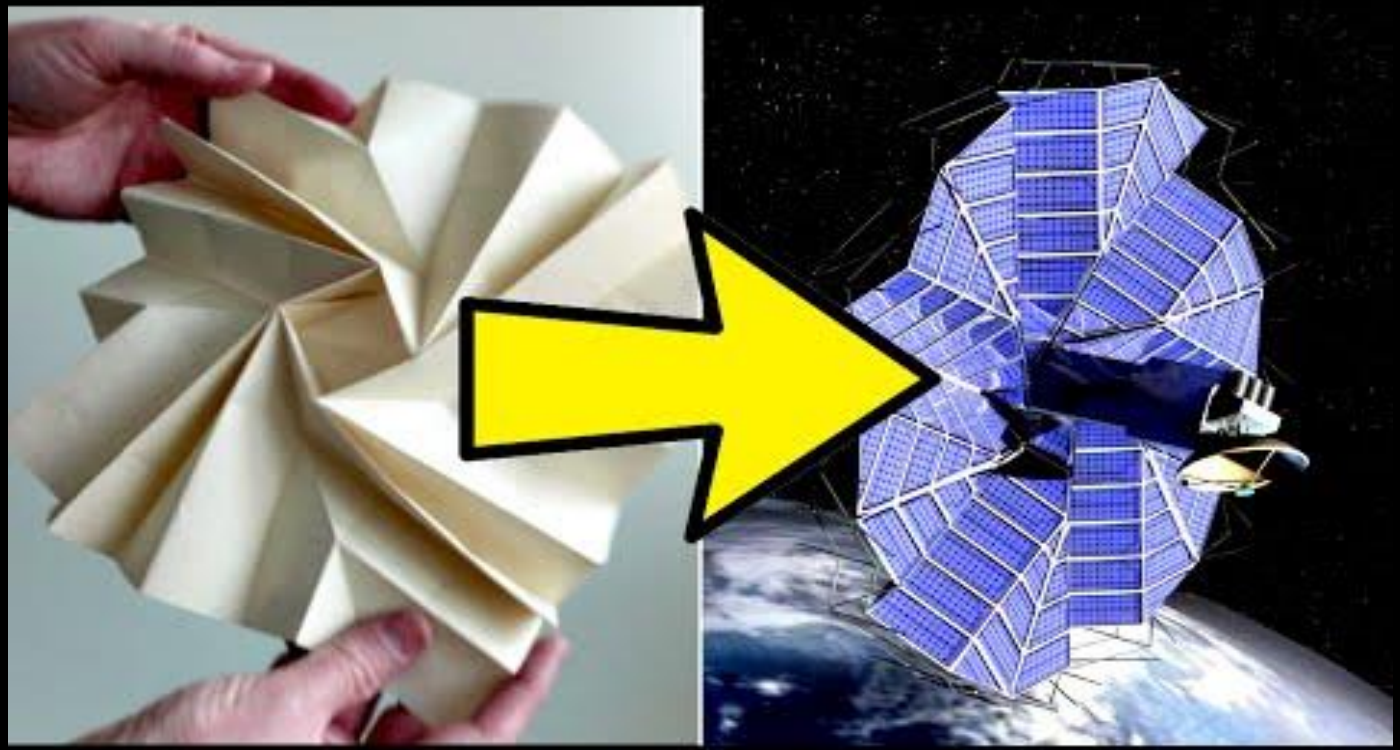
Given an *n*-ary relationship between entities $E_1, E_2, \ldots, E_n$:

- Identify relations $R_1, R_2, \ldots, R_n$ for entity types $E_1, E_2, \ldots, E_n$.
- Create a new relation R to represent the relationship.
- The primary key of R consists of the primary keys of $R_1, R_2, \ldots, R_n$.
- Create a foreign key in R to the primary key of each relation $R_1, R_2, \ldots, R_n$.
- Attributes of the relationship become attributes of R.


quantity
price
Part
pno {PK}
pname
desc
Provide
Project
jno {PK}
jname
budget
0..*
0..*
0..*
Supplier
sno {PK}
sname
address
Part (pno, pname, desc)
Project (jno, jname, budget)
Provide (pno, sno, jno, quantity, price)
Supplier (sno, sname, address)

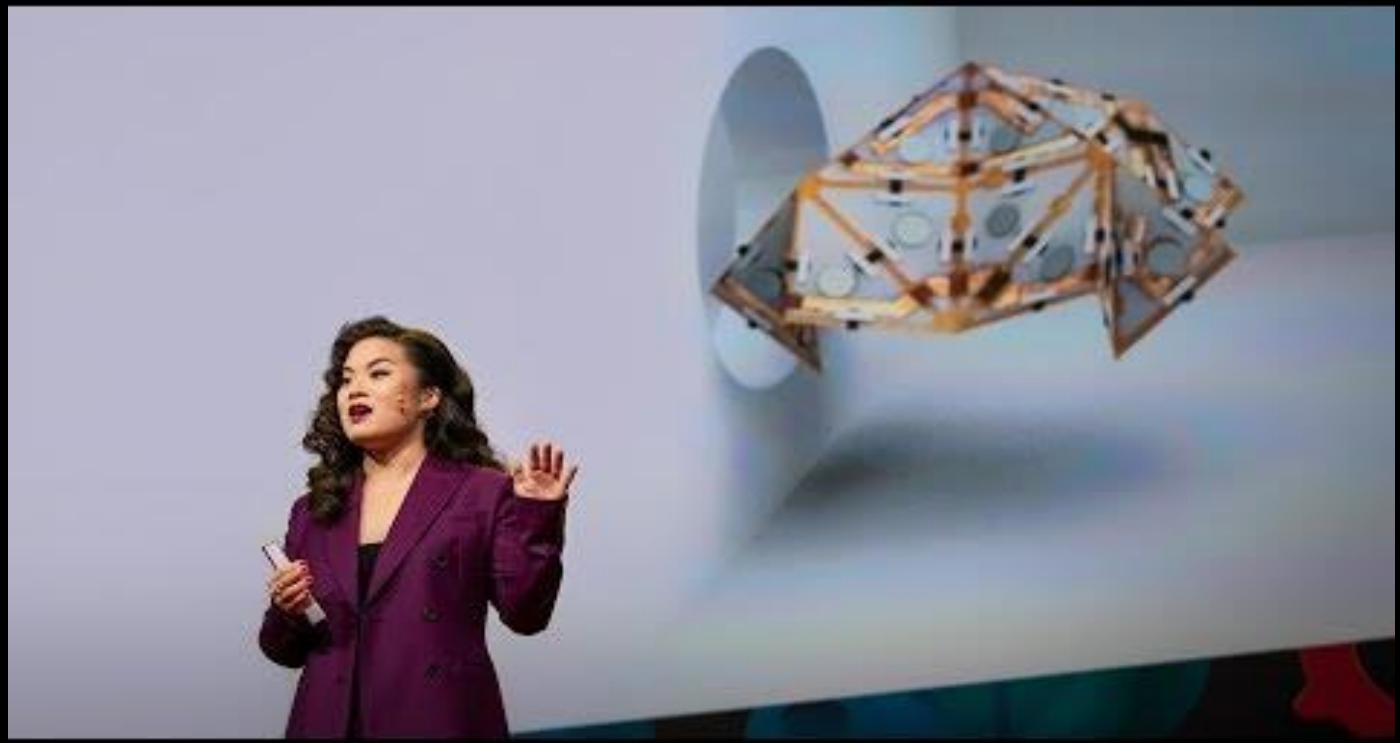
Time to Go Beyond the Course !!!

Engineering With Origami



Time to Go Beyond the Course !!!

Origami Robots



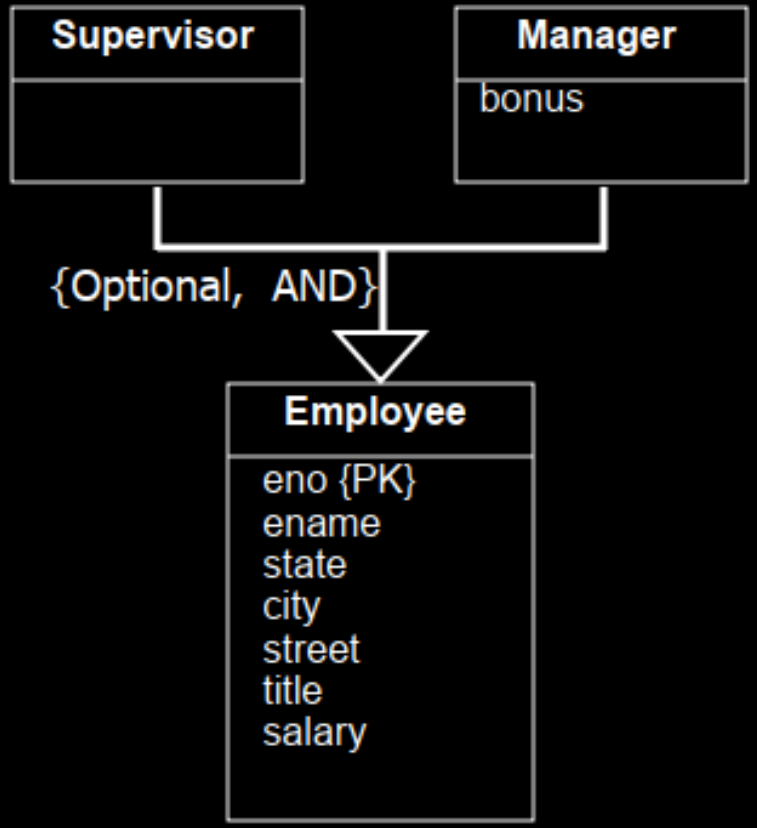
EER to Relational Mapping

An additional step to convert subclasses and superclasses to the relational model.

- 1) Create a separate relation for each superclass and subclass.
  - ⇨ Most general technique that we will use.

Convert subclasses and superclasses by creating a relation for each subclass and superclass. Link the subclasses to the superclass using foreign key references.

Supervisor (eno)



Manager (eno, bonus)

Employee (eno, ename, state, city, street, title, salary, supereno)

Alternatives

- 2) Create relations for subclass only.
  - ⇨ Only works if superclass has mandatory participation.
- 3) Create a single relation with one type attribute.
  - ⇨ Attribute is used to indicate the type of object (subclass) in the row.
  - ⇨ Works only if the subclasses are disjoint.
- 4) Create a single relation with multiple type attributes.
  - ⇨ Have a boolean valued attribute for each subclass. True if in subclass.
  - ⇨ Works if subclasses may be overlapping.

Concept Check !!!

How to deal with Entities?

Strong Entities

Weak Entities

How to deal with Attributes?

Simple

Composite

Multi-Valued

Key Attribute

How to deal with Relationships?

One to One

One to Many

Many to Many

How to deal with

Higher Degree Relationship

Functional Dependence & Normalization

Ideal Relational Schema

Minimize **Redundancy** & **Update Anomalies**

Redundancy occurs when the same data value is stored more than once in a relation.

- Redundancy wastes space and reduces performance.

Update anomalies are problems that arise when trying to insert, delete, or update tuples and are often caused by redundancy.

Universal Relation With All Attributes

Universal(eno, pno, resp, hours, ename, bdate, title, salary, supereno, dno, dname, mgreno, pname, budget)

eno	pno	resp	hours	ename	bdate	title	salary	supereno	dno	dname	mgreno	pname	budget
E1	P1	Manager	12	J. Doe	01-05-75	EE	30000	E2				Instruments	150000
E2	P1	Analyst	24	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5	Instruments	150000
E2	P2	Analyst	6	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5	DB Develop	135000
E3	P3	Consultant	10	A. Lee	07-05-66	ME	40000	E6	D2	Consulting	E7	Budget	250000
E3	P4	Engineer	48	A. Lee	07-05-66	ME	40000	E6	D2	Consulting	E7	Maintenance	310000
E4	P2	Programmer	18	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5	DB Develop	135000
E5	P2	Manager	24	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5	DB Develop	135000
E6	P4	Manager	48	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7	Maintenance	310000
E7	P3	Engineer	36	J. Jones	10-11-72	SA	50000		D1	Management	E8	Budget	250000

Duplicate values?

Challenges with Update?

Update Anomalies

There are three major types of update anomalies:

- ***Insertion Anomalies*** - Insertion of a tuple into the relation either requires insertion of redundant information or cannot be performed without setting key values to `NULL`.

- ***Deletion Anomalies*** - Deletion of a tuple may lose information that is still required to be stored.

- ***Modification Anomalies*** - Changing an attribute of a tuple may require changing multiple attribute values in other tuples.

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

Consider these two types of insertion anomalies:

- 1) Insert a new employee E9 working in department D2.
  - ⇨ You have to redundantly insert the department name and manager when adding this record.
- 2) Insert a department D4 that has no current employees.
  - ⇨ This insertion is not possible without creating a dummy employee id and record because `eno` is the primary key of the relation.

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

Consider this deletion anomaly:

- Delete employees E3 and E6 from the database.
- Deleting those two employees removes them from the database, and we now have lost information about department D2!

eno	ename	bdate	title	salary	supereno	dno	dname	mgreno
E1	J. Doe	01-05-75	EE	30000	E2	null	null	null
E2	M. Smith	06-04-66	SA	50000	E5	D3	Accounting	E5
E3	A. Lee	07-05-66	ME	40000	E7	D2	Consulting	E7
E4	J. Miller	09-01-50	PR	20000	E6	D3	Accounting	E5
E5	B. Casey	12-25-71	SA	50000	E8	D3	Accounting	E5
E6	L. Chu	11-30-65	EE	30000	E7	D2	Consulting	E7
E7	R. Davis	09-08-77	ME	40000	E8	D1	Management	E8
E8	J. Jones	10-11-72	SA	50000	null	D1	Management	E8

Consider these modification anomalies:

- 1) Change the name of department D3 to Embezzling.
  - ⇨ You must update the department name in 3 different records.
- 2) Change the manager of D1 to E4.
  - ⇨ You must update the `mgreno` field in 2 different records.

Normalization

Normalization is a technique for producing relations with desirable properties.

Normalization decomposes relations into smaller relations that contain less redundancy.

Decomposition requires that

no information is lost

reconstruction of the original relations must be possible.

Normalization

- Normalization can be used after ER modeling or independently.
- Normalization may be especially useful for databases that have already been designed without using formal techniques.

The purpose of normalization is to develop good relational schemas that minimize

redundancy and

update anomalies