Entity Relationship Modelling

P.J. McBrien

Imperial College London

Maintaining a Relational Database Schema

Designing a relational database schema

1. Describe the semantics of the UoD as a conceptual schema
   - ER (many variants exist)
   - UML class diagrams
2. Need to map the ER/UML schema into a relational schema: normalisation

Maintaining a Relational Database Schema

Designing a relational database schema

1. Describe the semantics of the UoD as a conceptual schema
   - ER (many variants exist)
   - UML class diagrams
2. Need to map the ER/UML schema into a relational schema: normalisation

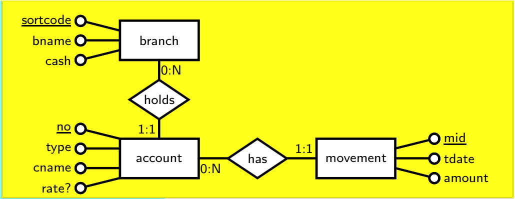
Maintaining a relational database schema

1. Want the semantics of the UoD as a conceptual schema
   - updates to schema
   - schema integration
2. If lost/never created, then need to reverse engineer the conceptual schema

Semantic Modelling: ER Schemas

```
CREATE TABLE branch
(     sortcode INTEGER NOT NULL,
      bname VARCHAR(20) NOT NULL,
      cash DECIMAL(10,2) NOT NULL,
      CONSTRAINT branch_pk PRIMARY KEY (sortcode)
)
CREATE TABLE account
(     no INTEGER NOT NULL,
      type CHAR(8) NOT NULL,
      cname VARCHAR(20) NOT NULL,
      rate DECIMAL(4,2) NULL,
      sortcode INTEGER NOT NULL,
      CONSTRAINT account_pk PRIMARY KEY (no),
      CONSTRAINT account_fk FOREIGN KEY (sortcode) REFERENCES branch
)

CREATE INDEX account_type ON account (type)
CREATE TABLE movement
(     mid INTEGER NOT NULL,
      no INTEGER NOT NULL,
      amount DECIMAL(10,2) NOT NULL,
      tdate DATETIME NOT NULL,
      CONSTRAINT movement_pk PRIMARY KEY (mid),
      CONSTRAINT movement_fk FOREIGN KEY (no) REFERENCES account
)
```



Core ER: Entities and Relationships

Entities

An entity E represents a set of objects which conceptually are the same type of thing

- **nouns** $\rightarrow$ entity set
- proper nouns imply instances, which are not entity sets.

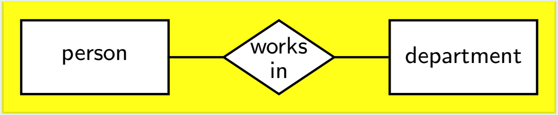
Relationships

A relationship R represents a set of tuples of objects where each tuple is some type of conceptual association between entities E_1, E_2

- **verbs** $\rightarrow$ relationship
- $R \subseteq \{\langle e_1, e_2 \rangle \mid e_1 \in E_1 \wedge e_2 \in E_2\}$

Identifying entities and relationships

In News Ltd, each person works in exactly one department; there are no restrictions on the number of persons a department may employ.



Core ER: Attributes of Entities

Attributes

An attribute $E.A$ is a function that maps from an entity set E to a value set V.

1. $E.A \subseteq \{\langle e, v\rangle | e \in E \wedge v \in V\}$
2. unique: $\langle e, v_1\rangle \in E.A \wedge \langle e, v_2\rangle \in E.A \rightarrow v_1 = v_2$
3. mandatory: $E = \{e \mid \langle e, v\rangle \in E.A\}$

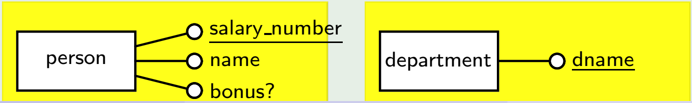
adjective, **adjective noun** $\rightarrow$ attribute

$\mathrm{ER}^{\mathcal{O}}$ and $\mathrm{ER}^{\mathcal{K}}$

- In $\mathrm{ER}^{\mathcal{O}}$: an **optional attribute** removes property (3)
- In $\mathrm{ER}^{\mathcal{K}}$: certain attribute(s) $E.A_1 \ldots E.A_n$ of E are denoted **key attributes** such that $E = \{\langle v_1, \ldots, v_n\rangle | \langle e, v\rangle \in E.A_1 \wedge \ldots \wedge \langle e, v_n\rangle \in E.A_n\}$

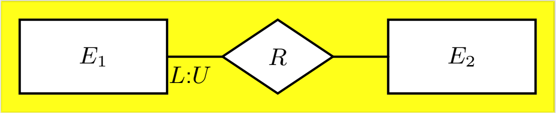
Identifying attributes

We record the name of each person working in the department; and identify them by their salary number. Optionally they might have a bonus figure recorded. Departments are identified by their name.



$ER^{\mathcal{L}}$: Look-Here Cardinality Constraints

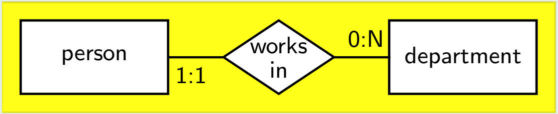
$ER^{\mathcal{L}}$



- An upper bound cardinality constraint U states that each instance of E_1 may appear at most U times in R. An upper bound of N indicates no limit.
- Additionally with $ER^{\mathcal{O}}$: a lower bound cardinality constraint L states that each instance of E_1 must appear at least L times in R

Adding look-here cardinality constraints in $ER^{\mathcal{LO}}$

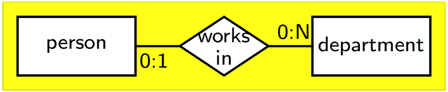
Each person works in exactly one department; there are no restrictions on the number of persons a department may employ.



Quiz 1: Extent of Relationships

person = {‘Peter’, ‘Jane’, ‘Mary’}

dept = {‘CS’, ‘Maths’}



Which is not a possible extent of works_in?

A

works_in={⟨‘Peter’,‘Maths’⟩, ⟨‘Peter’,‘CS’⟩, ⟨‘Mary’,‘Maths’⟩, ⟨‘Jane’,‘Maths’⟩}

B

works_in={⟨‘Peter’,‘Maths’⟩, ⟨‘Mary’,‘Maths’⟩, ⟨‘Jane’,‘Maths’⟩}

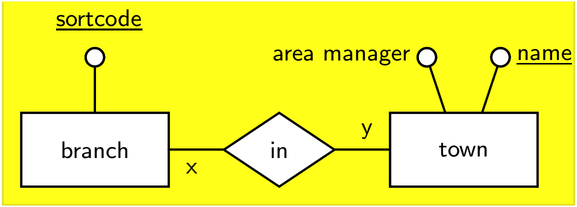
C

works_in={⟨‘Peter’,‘CS’⟩, ⟨‘Mary’,‘Maths’⟩, ⟨‘Jane’,‘Maths’⟩}

D

works_in={⟨‘Peter’,‘CS’⟩, ⟨‘Jane’,‘Maths’⟩}

Quiz 2: Cardinality Constraints on Relationships



Branches based in towns are all assigned to an area manager for that town; and area managers are only assigned to towns that have branches

What should be the cardinality constraints of in?

A
$x = 1{:}1, y = 0{:}N$

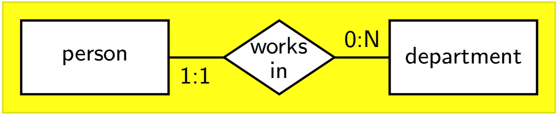
B
$x = 0{:}1, y = 0{:}N$

C
$x = 0{:}N, y = 1{:}N$

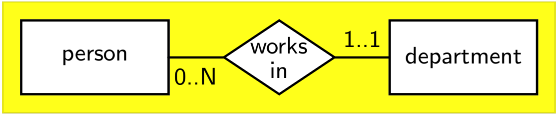
D
$x = 0{:}1, y = 1{:}N$

ER^C: Look-Across Cardinality Constraints

- This course uses **look-here** cardinality constraints: state the number of occurrences of the entity next to the constraint



- Other ER models use **look-across** cardinality constraints



$ER^{\mathcal{S}}$: Subset/isa hierarchies

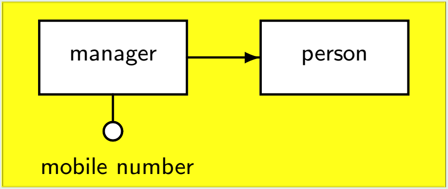
$ER^{\mathcal{S}}$

In $ER^{\mathcal{S}}$: if it is found that the instances of one entity E_s are a subset of a another entity E, we may add a **subset** constraint.
$E_s \subseteq E$

- **specialisation of nouns** → subset

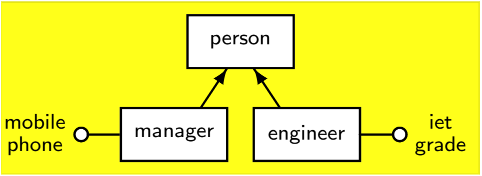
Identifying subsets with $ER^{\mathcal{S}}$

Some employees are ranked as managers, and receive a mobile phone.



Quiz 3: Extent of subset and superset entities

manager = {'Jane', 'Mary'}



Which is not a possible extent of person and engineer?

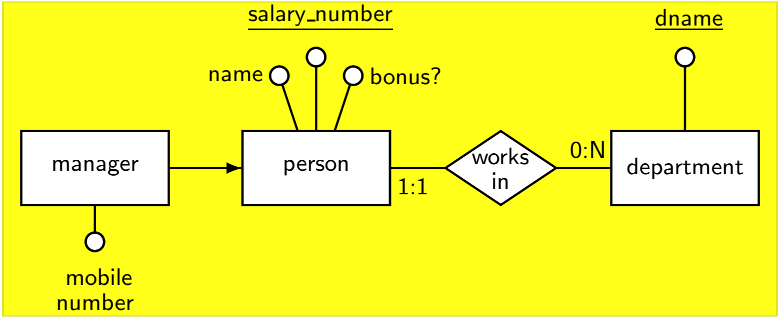
A
person={'Peter','Jane','Mary'}
engineer={'Jane','Mary'}

B
person={'Peter','Jane','Mary','John'}
engineer={}

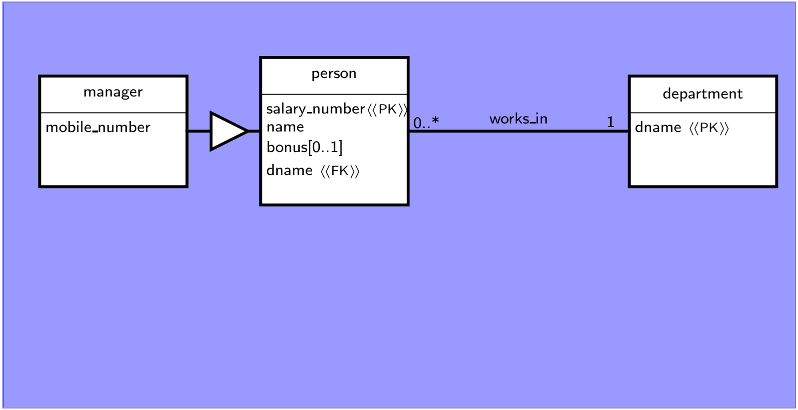
C
person={'Peter','Jane','Mary'}
engineer={'John'}

D
person={'Peter','Jane','Mary','John'}
engineer={'Peter','John'}

Combining Fragments



Using UML Class Diagrams as ER Models



How to Use UML Class Diagrams as an ER Schema

Use UML stereotypes to denote at least primary key information

Various approaches exist

ER Modelling Constructs $\mathcal{CKLOS}$

Construct	Description
$\mathcal{C}$	Look-across cardinality constraints
$\mathcal{L}$	Look-here cardinality constraints
$\mathcal{K}$	Key attributes
$\mathcal{O}$	Optional attributes
$\mathcal{S}$	Isa hierarchy between entities

You must choose between $\mathcal{C}$ or $\mathcal{L}$

Worksheet: ER Modelling

Draw an $\mathrm{ER}^{\mathcal{KLOS}}$ schema to describe the following domain

> *The payroll system for* BIG Inc *records the salaries, status, joining date, name, and payroll number for all of the corporation's 30,000 employees. Each employee works for one division, and each division has an account number for paying its staff. We identify divisions by their name, and record the address where the division's HQ is located.*
> *For employees sent abroad by* BIG Inc*, we record the address, country and telephone number of the foreign tax office that will handle the employee. It is assumed that each country has one central tax office that we have to deal with. All other employees have their tax affairs dealt with by the Inland Revenue.*

Worksheet: ER Modelling

Draw an $\text{ER}^{\mathcal{KLOS}}$ schema to describe the following domain

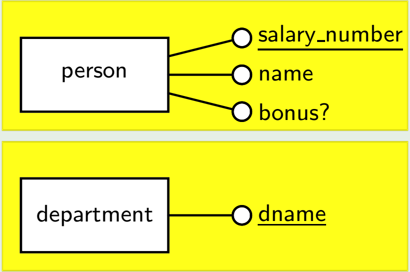
> *The payroll system for* BIG Inc *records the salaries, status, joining date, name, and payroll number for all of the corporation's 30,000 employees. Each employee works for one division, and each division has an account number for paying its staff. We identify divisions by their name, and record the address where the division's HQ is located.*
> *For employees sent abroad by* BIG Inc, *we record the address, country and telephone number of the foreign tax office that will handle the employee. It is assumed that each country has one central tax office that we have to deal with. All other employees have their tax affairs dealt with by the Inland Revenue.*

Mapping ER$^{\mathcal{KLOS}}$ to a relational model: entities and attributes

Taking a **table per type** (**TPT**) approach, there is a simple mapping of entities and attributes to tables and columns:

1. Each entity E maps to a table R_E
2. Each attribute A maps to a column C_A of R_E
3. If A is an optional attribute, then C_A is nullable, otherwise C_A is not nullable
4. If $\vec{K}$ are key attribute(s), then $\vec{C_K}$ are a key of R_E

Tables generated from entities



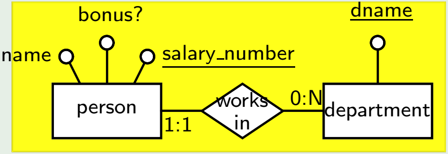
person(<u>salary_number</u>, name, bonus?)
department(<u>dname</u>)

Mapping ER$^{\mathcal{KLOS}}$ to a relational model: relationships

Taking a **table per type** (**TPT**) approach, for each relationship R between E_1, E_2, entities E_1, E_2 map to R_1, R_2 as before, and

1. If R is a many-many relationship then it maps to
    1. a table $R\_R_1\_R_2(\vec{K_1}, \vec{K_2})$
    2. a foreign key $R\_R_1\_R_2(\vec{K_1}) \stackrel{fk}{\Rightarrow} R_1(\vec{K_1})$
    3. a foreign key $R\_E_1\_E_2(\vec{K_2}) \stackrel{fk}{\Rightarrow} E_2(\vec{K_2})$
2. If R is a one-many relationship then it maps to
    1. a column $\vec{K_2}$ in R_1
    2. a foreign key $R_1(\vec{K_2}) \stackrel{fk}{\Rightarrow} R_2(\vec{K_2})$
    3. if the participation of E_1 in R is optional, then $\vec{K_2}$ is an optional column of R_1

Tables generated from relationships



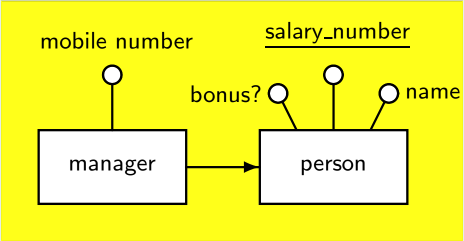
person(<u>salary_number</u>, name, bonus?, dname)
department(<u>dname</u>)
person(dname) $\stackrel{\text{fk}}{\Rightarrow}$ department(dname)

Mapping ER$^{\mathcal{KLOS}}$ to a relational model: subsets

Taking a **table per type** (**TPT**) approach, for each subset E_s of E, entities E_s, E map to tables R_s, R as before and:

1. a key $\vec{K}$ in R_s (where $\vec{K}$ is the key of R)
2. a foreign key $R_s(\vec{K}) \stackrel{fk}{\Rightarrow} R(\vec{K})$

Tables generated from subsets



person(salary_number, name, bonus?)
manager(salary_number, mobile_phone)
manager(salary_number) $\stackrel{fk}{\Rightarrow}$ person(salary_number)

Worksheet: Mapping ER$^{\mathcal{KLOS}}$ to a relational model

Take your ER$^{\mathcal{KLOS}}$ schema in the worksheet, and map it into a relational schema.

Relational ↔ ER

ER to relational mappings

- Design ER model for new DBMS system
- Map ER model to relations for DBMS implementation

Relational ↔ ER

ER to relational mappings

- Design ER model for new DBMS system
- Map ER model to relations for DBMS implementation

Relational to ER mappings

- Have existing DBMS that one wishes to view the design of
- Map relations to ER model for design review, modification and integration

Mapping Relational to ER$^{\mathcal{KLOS}}$: (Assuming TPT)

1. If table R has just primary keys P_R that contains two columns which are foreign keys for R_1, R_2 $\rightarrow$ many-many ER relationship R between entities for R_1, R_2
2. Otherwise $R \rightarrow$ ER entity
   For each column A:

Mapping Relational to ER$^{\mathcal{KLOS}}$: (Assuming TPT)

1. If table R has just primary keys P_R that contains two columns which are foreign keys for R_1, R_2 $\rightarrow$ many-many ER relationship R between entities for R_1, R_2
2. Otherwise $R \rightarrow$ ER entity
   For each column A:
   1. If A is (part of) a candidate key:
      1. if candidate key is also a foreign key $\rightarrow$ subset
      2. otherwise $A \rightarrow$ ER key attribute

Mapping Relational to ER$^{\mathcal{KLOS}}$: (Assuming TPT)

1. If table R has just primary keys P_R that contains two columns which are foreign keys for R_1, R_2 → many-many ER relationship R between entities for R_1, R_2
2. Otherwise R → ER entity
   For each column A:
   1. If A is (part of) a candidate key:
      1. if candidate key is also a foreign key → subset
      2. otherwise A → ER key attribute
   2. If A is not (part of) a candidate key:
      1. if A is (part of) a foreign key → ER relationship
      2. otherwise A → ER non-key attribute

Mapping Relational to ER$^{\mathcal{KLOS}}$ using TPT: Example

site(sitecode,sortcode?,name)

withdraw(sitecode,cname)

customer(cname,joined,salary?,address,phone)

web_customer(cname,username,password,email)

account(number,acname,cname,sitecode)

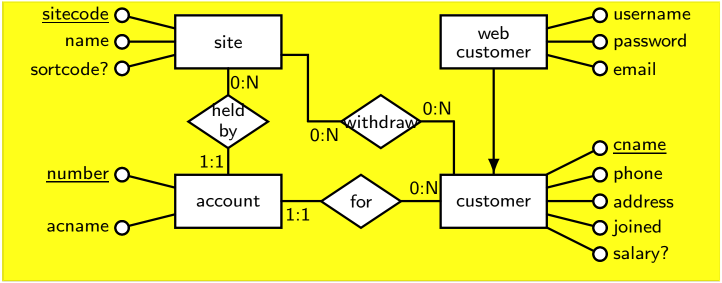
withdraw(sitecode) $\stackrel{fk}{\Rightarrow}$ site(sitecode)

withdraw(cname) $\stackrel{fk}{\Rightarrow}$ customer(cname)

web_customer(cname) $\stackrel{fk}{\Rightarrow}$ customer(cname)

account(cname) $\stackrel{fk}{\Rightarrow}$ customer(cname)

account(sitecode) $\stackrel{fk}{\Rightarrow}$ site(sitecode)



Worksheet: Reverse Engineering ER Schemas

Build an $\mathrm{ER}^{\mathcal{KLOS}}$ schema representing the following relational schema

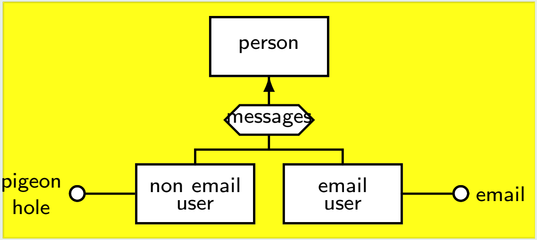
person(<u>name</u>,dcode,salary,age?)
person(dcode) $\stackrel{fk}{\Rightarrow}$ department(dcode)
manager(<u>name</u>,dcode,car)
manager(name) $\stackrel{fk}{\Rightarrow}$ person(name)
department(<u>dcode</u>,site)
sales_department(<u>dcode</u>,telephone)
sales_department(dcode) $\stackrel{fk}{\Rightarrow}$ department(dcode)
production_department(<u>dcode</u>)
production_department(dcode) $\stackrel{fk}{\Rightarrow}$ department(dcode)
department_handles_product(<u>dcode,pcode</u>)
department_handles_product(dcode) $\stackrel{fk}{\Rightarrow}$ department(dcode)
department_handles_product(pcode) $\stackrel{fk}{\Rightarrow}$ product(pcode)
product(<u>pcode</u>,price,weight)

ER$^{\mathcal{D}}$: Disjointness and Generalisation Hierarchies

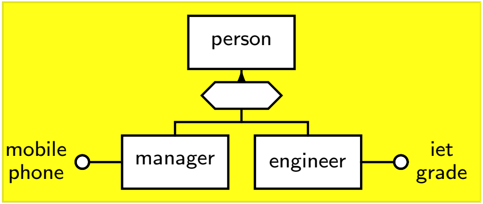
- In ER$^{\mathcal{D}}$: the disjointness of entities $E_1 \ldots E_n$ may be specified, enforcing that $\forall x, y . x \neq y \rightarrow E_x \cap E_y = \emptyset$
- The notion of **generalisation hierarchies** combines the use of disjointness and subset.
- **disjoint specialisation of nouns** $\rightarrow$ generalisation

Identifying generalisation hierarchies in ER$^{\mathcal{SD}}$

Employees may also be divided, according to how they like to receive messages, into email users and non-email users. The former must have a email address recorded, the later must have a pigeon hole number recorded.



Quiz 4: Extent of generalisation entities



Which is not a possible extent the entities?

A

person={'Peter','Jane','Mary'}
engineer={'Peter','John'}
manager={'Jane','Mary'}

B

person={'Peter','Jane','Mary','John'}
engineer={}
manager={'Jane','Mary'}

C

person={'Peter','Jane','Mary','John'}
engineer={'John'}
manager={'Jane','Mary'}

D

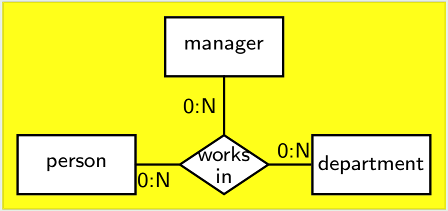
person={'Peter','Jane','Mary','John'}
engineer={'Peter','John','Mary'}
manager={'Jane','Mary'}

$\text{ER}^{\mathcal{H}}$: Allowing an n-ary relationship

- In graph theory, an edge connecting more that two nodes is called a **hyper-edge**.
- In $\text{ER}^{\mathcal{H}}$: allow n-ary relationships between entities, rather than just binary
- An n-ary relationship is equivalent to a weak entity with n binary relationships

Identifying an n-ary relationship

A person may work in multiple departments, and for each department the person works in, the person will be assigned a manager

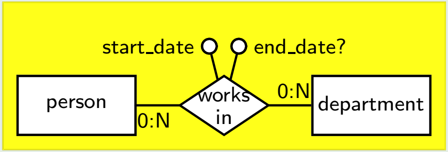


$ER^{\mathcal{A}}$: Allowing attributes on relationships

- Use when there are values to be associated with the relationship between entities

Identifying an attribute of a relationship

We record the ***start_date*** *when a person joined a department, and when the person leaves, record the* ***end_date*** *they left the department. We keep a history of all departments the person worked in.*



Mapping Relational to ER$^{\mathcal{AHKLOS}}$ (Assuming TPT)

1. For each table R that has primary key P_R that contains n columns which are foreign keys for $R_1, \ldots, R_n$ $(n \geq 2) \rightarrow$ many-many ER relationship R between entities for $R_1, \ldots, R_n$. For each column A of R:
    1. If column part of primary key, then already represented by many-many relationship
    2. Otherwise, create attribute A of relationship
2. Otherwise each table $R \rightarrow$ ER entity. For each column A on R.
    1. If A is (part of) a candidate key:
        1. if candidate key is also a foreign key $\rightarrow$ subset
        2. otherwise $A \rightarrow$ ER key attribute
    2. If A is not (part of) a candidate key:
        1. if A is (part of) a foreign key $\rightarrow$ ER relationship
        2. otherwise $A \rightarrow$ ER non-key attribute
3. Use additional domain knowledge to convert several subclasses of a single entity into members of a single generalisation hierarchy

Alternatives for mapping ER to Relational Form

- Table per Type (TPT)
  - Generates a large number of tables
  - Does not implement disjointness in subclasses (unless triggers are used . . .)

Alternatives for mapping ER to Relational Form

- Table per Type (TPT)
  - Generates a large number of tables
  - Does not implement disjointness in subclasses (unless triggers are used . . .)
- Table per Concrete Class (TPC)
  - Generates fewer tables, but with many nullable attributes
  - Implement disjointness in subclasses, but does not implement mandatory attributes or relationships (unless CHECK constraints are used)

Alternatives for mapping ER to Relational Form

- Table per Type (TPT)
  - Generates a large number of tables
  - Does not implement disjointness in subclasses (unless triggers are used ...)
- Table per Concrete Class (TPC)
  - Generates fewer tables, but with many nullable attributes
  - Implement disjointness in subclasses, but does not implement mandatory attributes or relationships (unless CHECK constraints are used)
- Table per Hierarchy (TPH)
  - Generates even fewer tables, superclass instances now spread between subclass tables.
  - Does not implement disjointness in subclasses (unless triggers are used), but implement mandatory attributes or relationships.

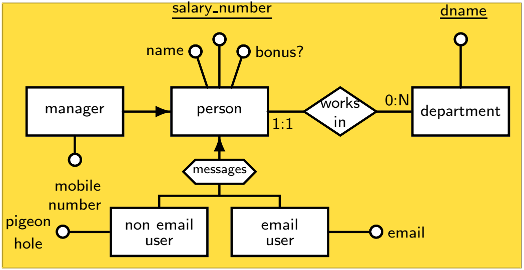
Mapping ER to Relational Form (Explicit Inheritance)

- Each entity maps to a distinct table, except superclass entities of total generalisations, which disappear!
- Each attribute maps to a column
- Each relationship
  - that is many-many maps to a table
  - that is one-many maps to a column in the table of the 'one' end and a foreign key pointing at the many table, provided the many table has no subclasses
- Each isa and generalisation causes the attributes of superclass entity to also appear as columns of the subclass table

Table per Concrete Type (TPC)

In ORM, this approach is called table per concrete type, since each non-virtual class type maps into a table, with the variables of super classes in the table.

Mapping ER to Relational Form (Explicit Inheritance)



manager(salary_number,name,bonus?,dname,mobile_number)
manager.dname $\stackrel{fk}{\Rightarrow}$ department.dname
department(dname)
non_email_user(salary_number,name,bonus?,dname,pigeon_hole)
non_email_user.dname $\stackrel{fk}{\Rightarrow}$ department.dname
email_user(salary_number,name,bonus?,dname,email)
email_user.dname $\stackrel{fk}{\Rightarrow}$ department.dname

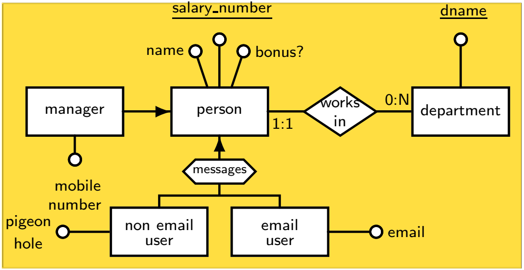
Mapping ER to Relational Form (ISA as Attribute)

- Each non-subclass entity maps to a distinct table
- Each attribute maps to a column in the superclass table (becoming nullable if moved up from a subclass)
- Each relationship
  - that is many-many maps to a table
  - that is one-many maps to a column in the table of the 'one' end and a foreign key pointing at the many table
- Each isa maps to a boolean flag
- Each generalisation maps to an attribute taking enumerated values

Table per Hierarchy (TPH)

In ORM, this approach is called table per hierarchy, since each class hierarchy maps into a single table

Mapping ER to Relational Form (ISA as Attribute)



person(<u>salary_number</u>,name,bonus?,dname,
is_manager,mobile_number?,
messages,pigeon_hole?,email?)

person.dname $\overset{fk}{\Rightarrow}$ department.dname

department(<u>dname</u>)