Mapping dal modello concettuale (ER) al modello logico (relazionale)

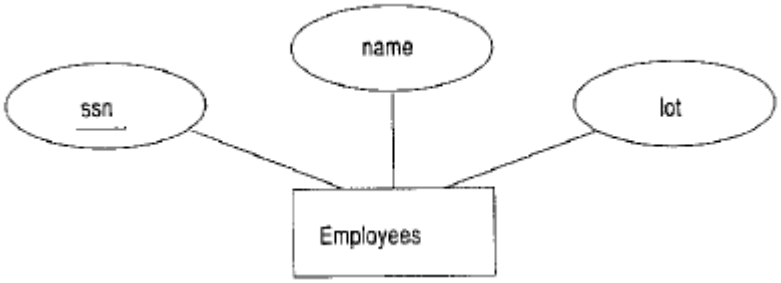
Obiettivi del mapping

- Preservare il più possibile la conoscenza rappresentata nel modello concettuale
- Mantenere i vincoli (per quanto possible)
  - Alcuni vincoli non sono esprimibili nel modello relazionale:
    - Cardinalità diverse da 1 ed N;
    - Vincoli di partecipazione per relazioni con grado > 2
- Minimizzare i valori NULL

Le procedure di mapping qui presentate sono implementate in molti tool commerciali

Da Entity Type a Tabelle

- Per ogni Entity type E nello schema ER, creare una tabella R che ha come attributi tutti gli attributi di E
- Porre uno degli attributi chiave di E come chiave primaria per R.
- Se la chiave di E è composta, la chiave di R sarà l'insieme dei corrispettivi attributi di R



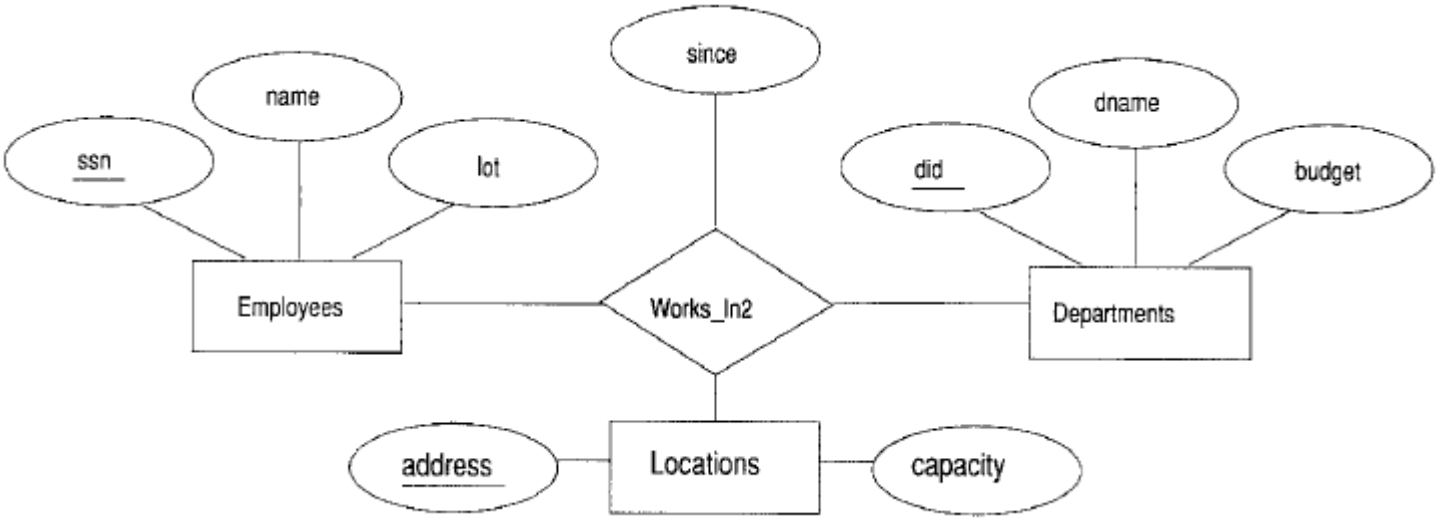
ssn	*name*	*lot*
123-22-3666	Attishoo	48
231-31-5368	Smiley	22
131-24-3650	Smethurst	35

```
CREATE TABLE Employees ( ssn      CHAR(11),
                         name     CHAR(30),
                         lot      INTEGER,
                         PRIMARY KEY (ssn) )
```

Mapping da relazioni (senza vincoli)

- Data una relazione (nel senso di ER) tra due Entity Types, creare una relazione con i seguenti attributi:
  - La chiave primaria di ogni entità partecipante (come chiave esterna)
  - Gli attributi descrittivi della relazione
- Gli attributi non descrittivi di questa nuova relazione sono una superchiave (se non ci sono vincoli di chiave in ER, sono una chiave candidata)

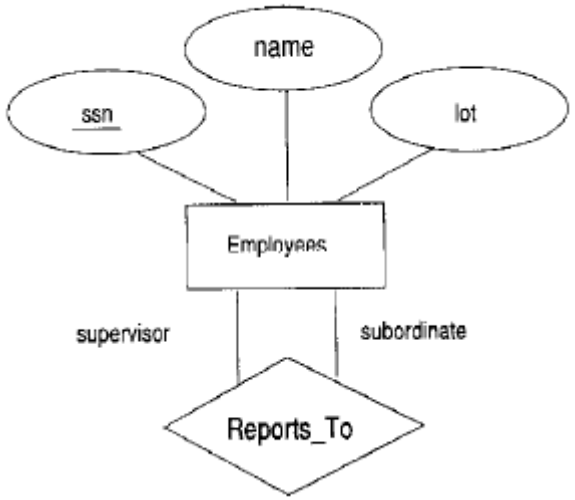
Mapping da relazioni (senza vincoli)



ssn	did	address	since
…	…	…	…
…	…	…	…

```
CREATE TABLE Works_In2 ( ssn       CHAR(11),
                         did       INTEGER,
                         address   CHAR(20),
                         since     DATE,
                         PRIMARY KEY (ssn, did, address),
                         FOREIGN KEY (address) REFERENCES Locations,
                         FOREIGN KEY (did) REFERENCES Departments )
```

Mapping da relazioni: le relazioni ricorsive



supervior_ssn	subordinate_ssn
...	...
...	...

```
CREATE TABLE Reports_To (
        supervisor_ssn   CHAR(11),
        subordinate_ssn  CHAR(11),
        PRIMARY KEY (supervisor_ssn, subordinate_ssn),
        FOREIGN KEY (supervisor_ssn) REFERENCES Employees(ssn),
        FOREIGN KEY (subordinate_ssn) REFERENCES Employees(ssn) )
```

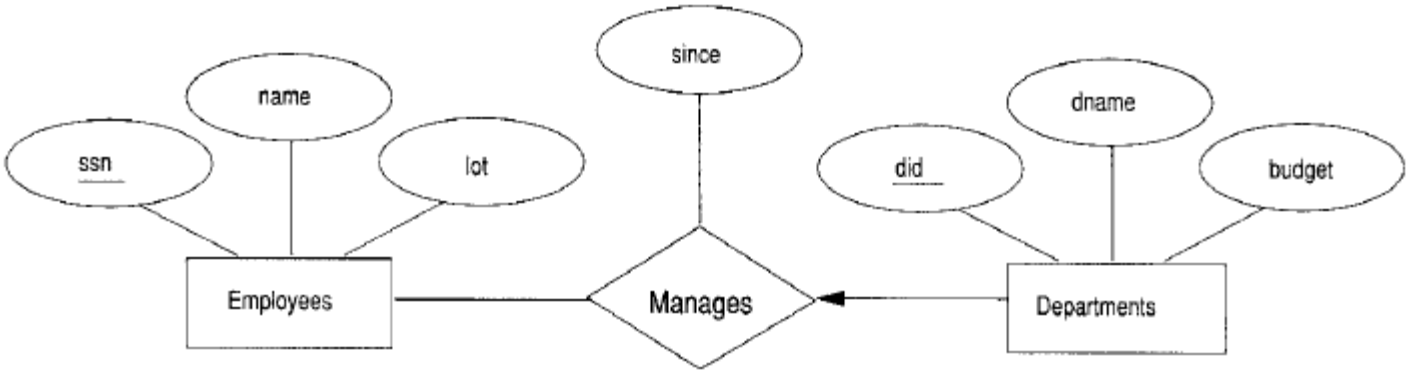
Mapping da relazioni: vincolo di chiave

- Prendiamo due entità E_1 ed E_2 legate da una relazione R e supponiamo che su E_2 esista un vincolo di chiave.

- Il mapping di base non va bene, perché solo i valori di PK_2 non possono ripetersi.
  - La coppia $<PK_1, PK_2>$ quindi è una superchiave, quindi non può essere una chiave candidata

- Due soluzioni possibili:
  A. Creare una nuova relazione R_A che ha come attributi PK_1 e PK_2 (chiavi esterne) ma solo PK_2 come chiave primaria + eventuali attributi descrittivi
  B. Creare una nuova relazione R_B ottenuta aggiungendo a E_2 un nuovo attributo (chiave esterna) che referenzia la chiave primaria di E_1 (sfruttando il fatto che comunque il valore di PK_2 non può ripetersi a causa del vincolo di chiave) + eventuali attributi descrittivi

Mapping da relazioni: vincolo di chiave



A

```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

B

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```

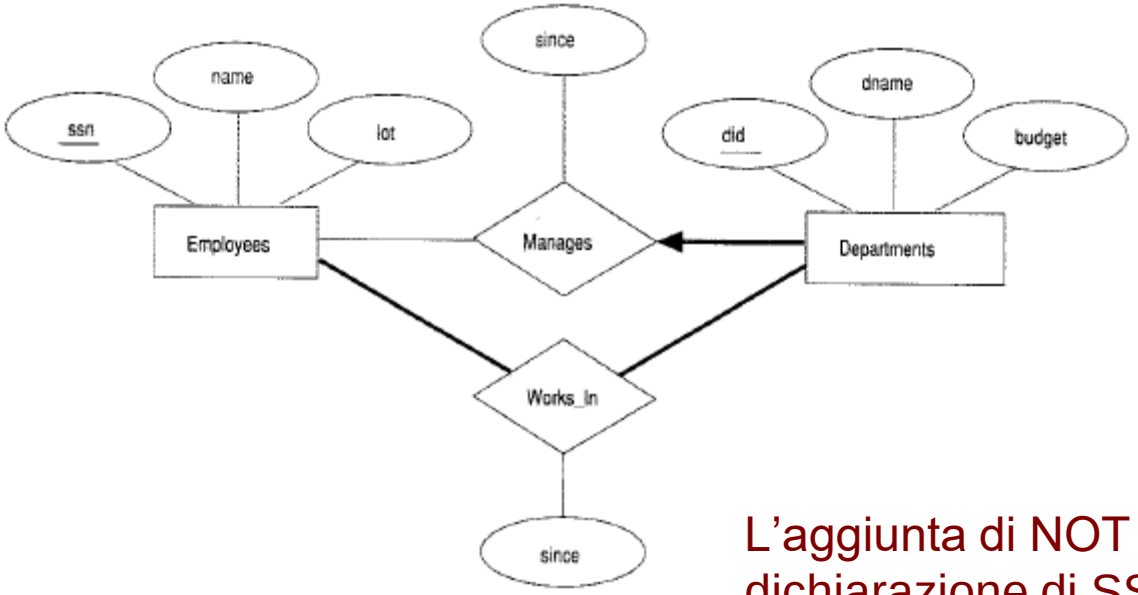
Mapping da relazioni: vincolo di chiave

- Pro e contro delle due soluzioni:
  - Soluzione B:
    - PRO: elimina la necessità di una nuova relazione MANAGES e permette di associare un dipartimento al suo manager senza dover unire attributi di tabelle diverse
    - CONS: se ci sono molti dipartimenti senza manager avremo tanti NULL come valore di SSN
  - Soluzione A:
    - PRO: elimina il problema dei NULL
    - CONS: molte operazioni possono richiedere di unire attributi di tabelle diverse
- Questi approcci possono essere estesi a relazioni di grado > 2

Mapping da relazioni: vincolo di partecipazione

- Il vincolo di **partecipazione parziale** non richiede alcuna specifica particolare, corrisponde all'assenza di vincoli
- Il vincolo di **partecipazione totale** non può essere rappresentato in modo completo (senza perdita di informazioni rispetto al modello ER) e può essere catturato solo con una combinazione di specifiche del modello relazionale e l'utilizzo di altri strumenti del linguaggio SQL (*table constraints* e *assertions*)
- Nell'esempio che segue vedremo come catturare almeno un aspetto del vincolo di partecipazione
- Rimandiamo la discussione degli altri aspetti a quando esamineremo il linguaggio SQL

Mapping di relazioni: vincolo di partecipazione




```
CREATE TABLE Dept_Mgr ( did      INTEGER,
                        dname    CHAR(20),
                        budget   REAL,
                        ssn      CHAR(11) NOT NULL,
                        since    DATE,
                        PRIMARY KEY (did),
                        FOREIGN KEY (ssn) REFERENCES Employees
                                ON DELETE NO ACTION )
```

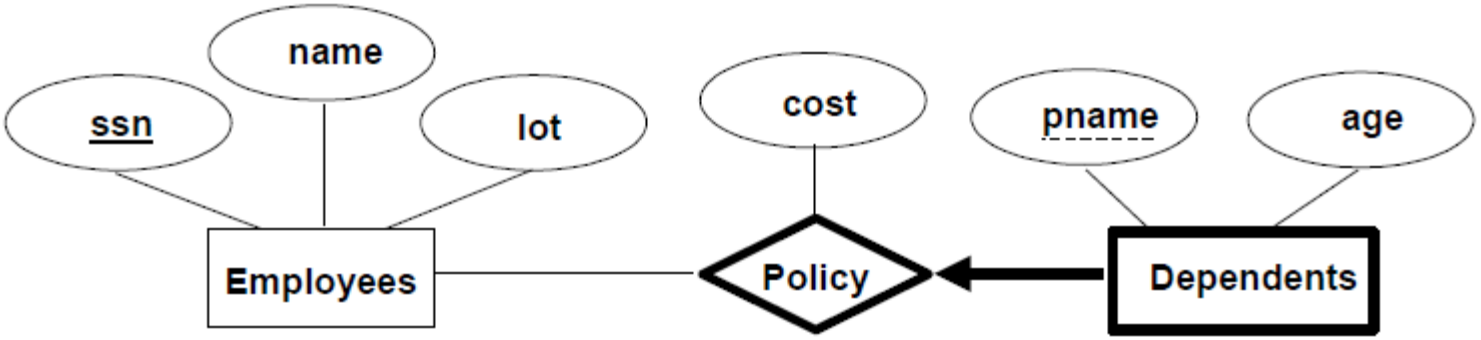
L'aggiunta di NOT NULL nella dichiarazione di SSN ha come effetto che non possa esserci un dipartimento per il quale non è specificato un manager, quindi **ogni** dipartimento ha un manager

Mapping di relazioni: *weak entities*

- Le entità deboli:
  - partecipano sempre a una relazione 1:N con l'entità identificante
  - hanno un vincolo di chiave
  - hanno un vincolo di partecipazione totale

- Le due caratteristiche di cui tenere conto sono:
  - hanno solo una chiave parziale
  - devono essere cancellate se viene cancellata l'entità da cui dipendono

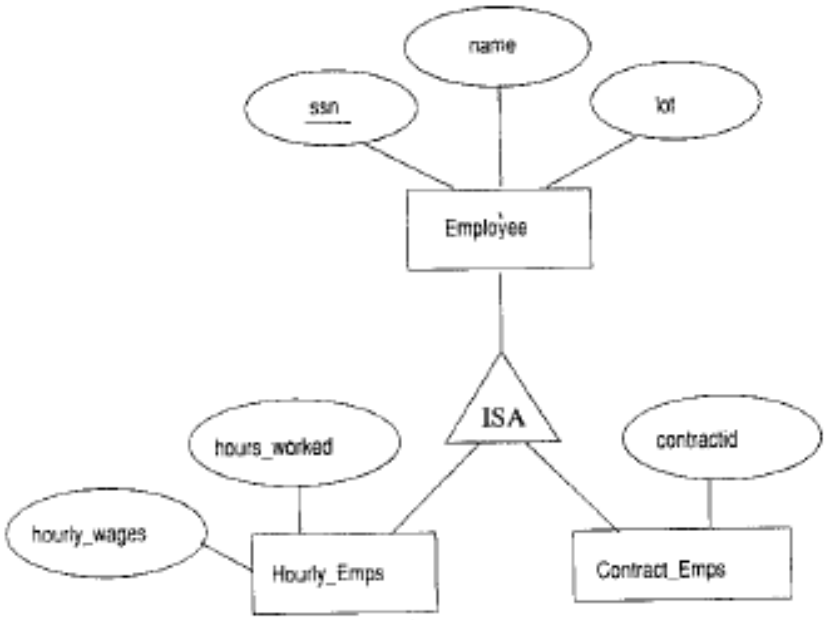
Mapping di relazioni: *weak entities*




```
CREATE TABLE Dep_Policy (
  pname CHAR(20),
  age INTEGER,
  cost REAL,
  ssn CHAR(11) NOT NULL,
  PRIMARY KEY (pname, ssn),
  FOREIGN KEY (ssn) REFERENCES Employees,
    ON DELETE CASCADE)
```

Mapping di gerarchie IS-A

- Assumiamo di dover mappare il seguente diagramma ER



Mapping di gerarchie IS-A

- Ci sono due principali approcci per il mapping:

  1. Generale: creare 3 diverse entità, una per ogni classe del diagramma ER

  2. Eliminare l'entità più generale (EMPLOYEES) e limitarsi a mappare le due sottoclassi

- Vediamoli separatamente

Mapping di gerarchie IS-A

- Metodo generale:
  - Creare la relazione EMPLOYEES come al solito
  - Creare la relazione HOURLY_EMPS con attributo ssn come chiave primaria e chiave esterna che referenzia ssn in EMPLOYEES:

```
HOURLY_EMPS(hourly_wages, hours_worked, ssn)
```

  - Creare la relazione CONTRACT_EMPS in modo analogo
  - Aggiungere l'opzione ON DELETE CASCADE per assicurarsi che le tuple che referenziano una tupla in EMPLOYEES sia cancellata quando la tupla a cui fa riferimento in EMPLOYEES viene cancellata

Mapping di gerarchie IS-A

- Metodo di riduzione:
  - Creare solo le relazioni HOURLY_EMPS e CONTRACT_EMPS, aggiungendo a ognuna di esse gli attributi che ereditati da EMPLOYEES nel diagramma ER:

HOURLY_EMPS(ssn, name, lot, hourly_wages, hours_worked)

CONTRACT_EMPS(ssn, name, lot, contract_id)

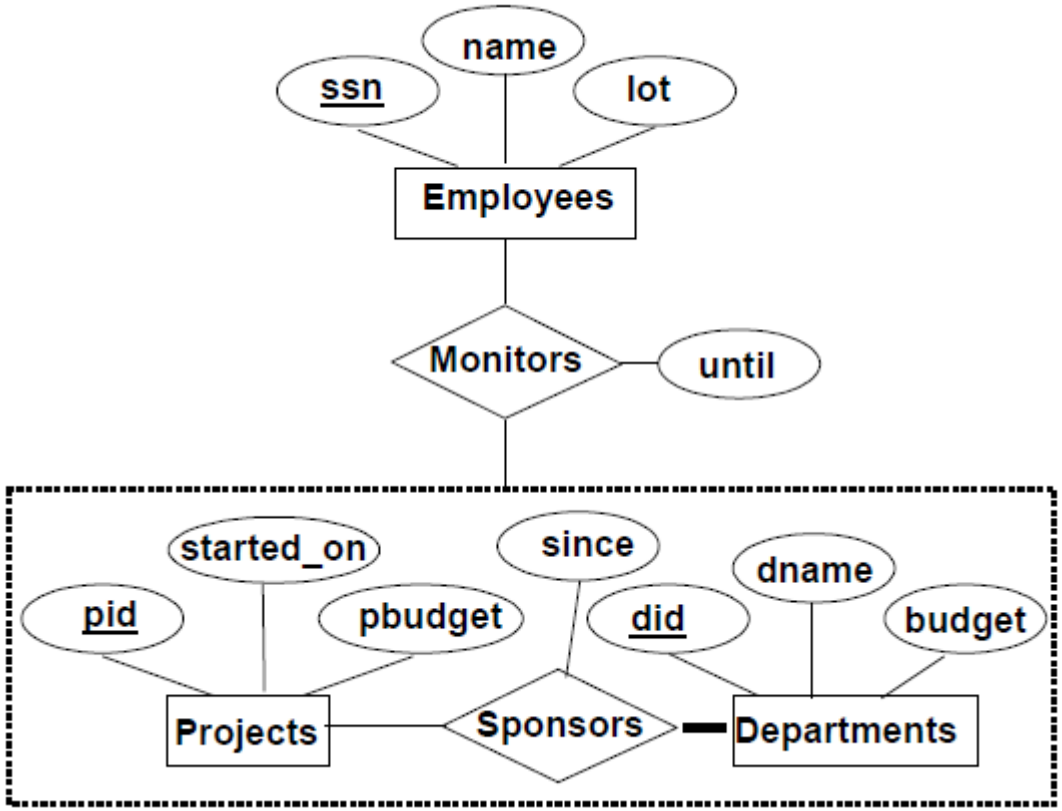
  - In questa soluzione, tutti gli impiegati devono essere compresi in una delle due relazioni (la copertura deve essere totale)

Mapping di gerarchie IS-A

- Confronto tra i due metodi:
  - Il metodo generale:
    - si può applicare in ogni situazione
    - le query che riguardano gli attributi di EMPLOYEES non richiedono di esaminare le altre due relazioni
    - le query che riguardano gli attributi specifici delle due sottoclassi possono richiedere di combinare più relazioni per ottenere, ad esempio, il valore di *name* o *lot* di un impiegato a ore
  - Il metodo di riduzione:
    - non si applica se ci sono impiegati che non appartengono a nessuna delle due sotto-classi
    - se c'è *overlapping* (un impiegato può appartenere a entrambe le relazioni), le informazioni su *name* e *lot* sono duplicate nelle due tabelle
    - Le query che riguardano tutti gli impiegati devono esaminare entrambe le relazioni
- La scelta tra i due metodi dipende quindi dalla semantica dei dati e dalla frequenza con cui vengono eseguite specifiche query

Mapping di aggregazioni

- Partiamo dall'esempio visto in precedenza



Mapping di aggregazioni

- Approccio generale:
  - Le entità EMPLOYEES, PROJECTS e DEPARTMENT sono mappate come al solito
  - Anche la relazione SPONSORS è mappata come visto in precedenza
  - Per la relazione MONITORS, creiamo una nuova tabella con il seguente schema (contenente le chiavi delle altre entità e gli eventuali attributi descrittivi della relazione):

```
MONITORS(ssn, did, pid, until)
```

Mapping di aggregazioni

- C'è un caso speciale in cui il mapping può essere semplificato, ovvero quando:
  - La relazione esterna all'aggregazione (SPONSORS nel nostro esempio) non ha attributi descrittivi
  - L'aggregazione ha un vincolo di partecipazione totale in MONITORS (ovvero, ogni coppia progetto-dipartimento **deve** avere un impiegato che la controlla)

- Se queste condizioni valgono, la relazione SPONSORS può essere rimossa dal mapping, visto che tutte le sue istanze possono essere ricavate dalle colonne <*pid*, *did*> della relazione MONITORS