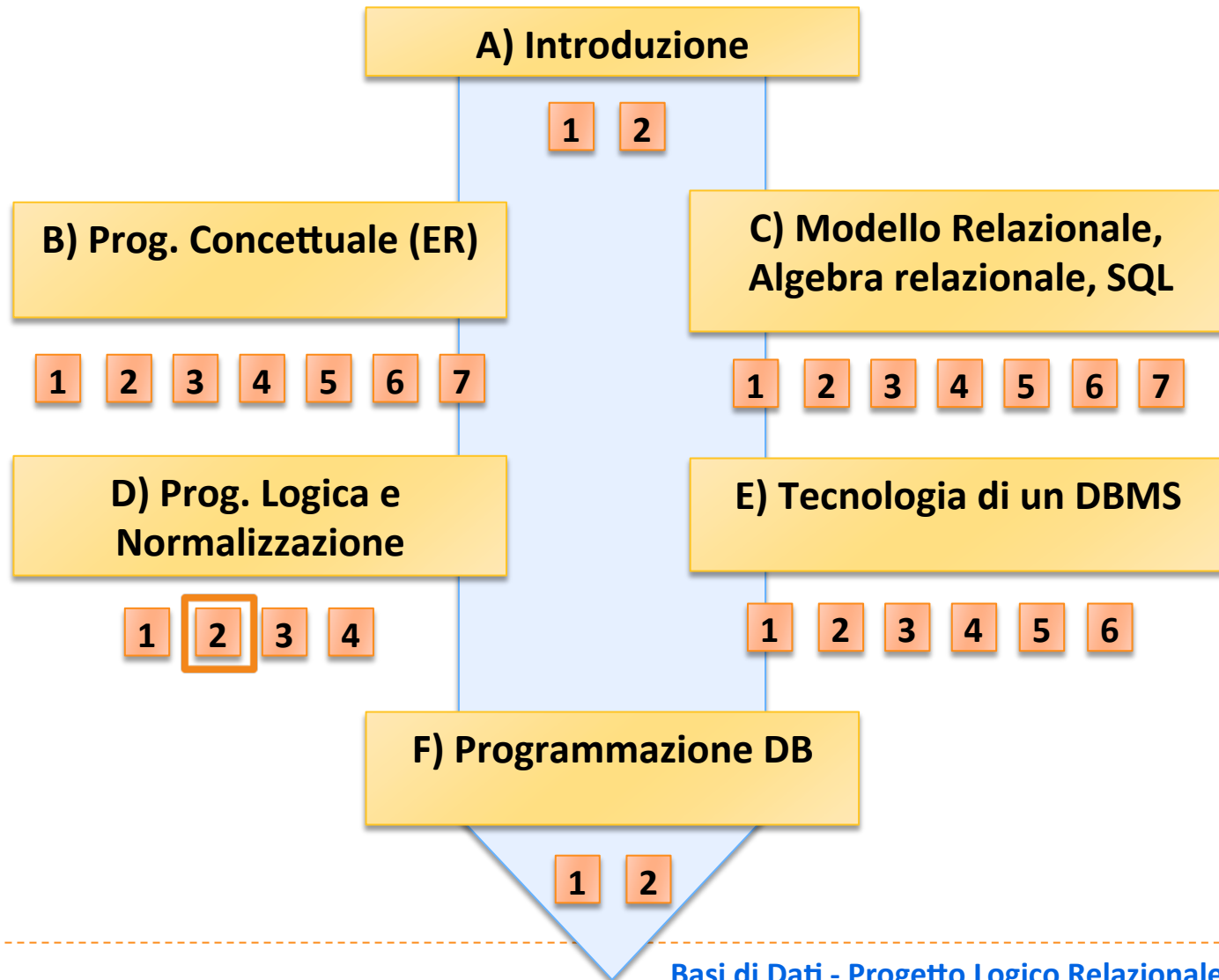


Basi di Dati

Progetto Logico Relazionale (Parte 2)

Basi di Dati – Dove ci troviamo?



Altre traduzioni

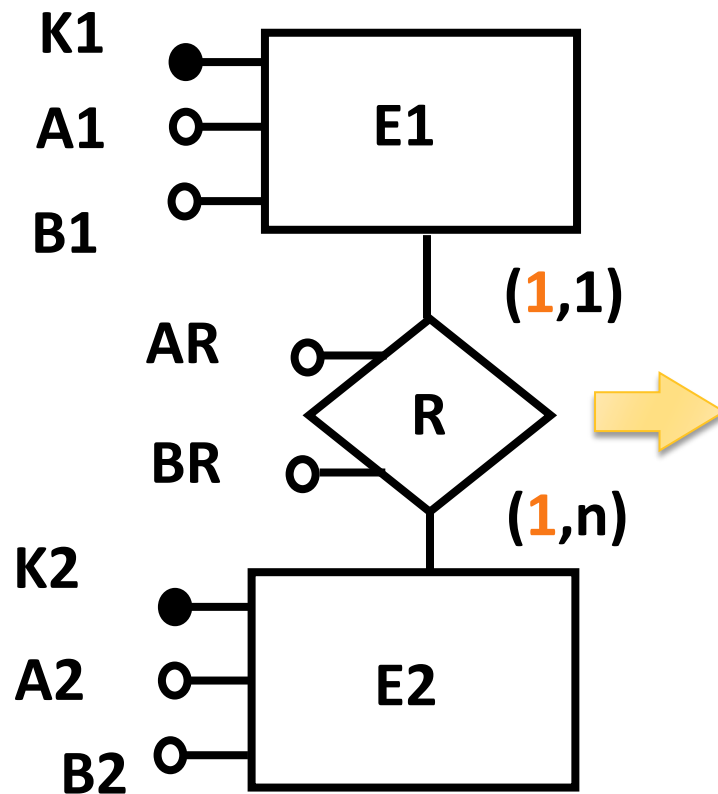
- ▶ La **traduzione standard** è **sempre possibile** ed è l'**unica** possibilità per le associazioni N a M
- ▶ Altre forme di traduzione delle associazioni sono possibili **per altri casi di cardinalità** (1 a 1, 1 a N)
- ▶ Le altre forme di traduzione **fondono** in una stessa relazione entità e associazioni

Altre traduzioni

- ▶ **Le altre forme di traduzione:**
 - ▶ danno origine a un **minor numero** di relazioni e generano quindi uno schema più semplice
 - ▶ richiedono un minor numero di **join** per la **navigazione** attraverso un'associazione, ovvero per accedere alle istanze di entità connesse tramite l'associazione
 - ▶ penalizzano le operazioni che consultano **soltanto** gli attributi di una entità che è stata fusa

Associazione binaria 1 a N

► traduzione standard:



E1 (K1, A1, B1)

E2 (K2, A2, B2)

R (K1, K2, AR, BR)

Associazione binaria 1 a N

- ▶ Se E1 partecipa con cardinalità (1,1) può essere fusa con l'associazione, ottenendo una soluzione a due relazioni:

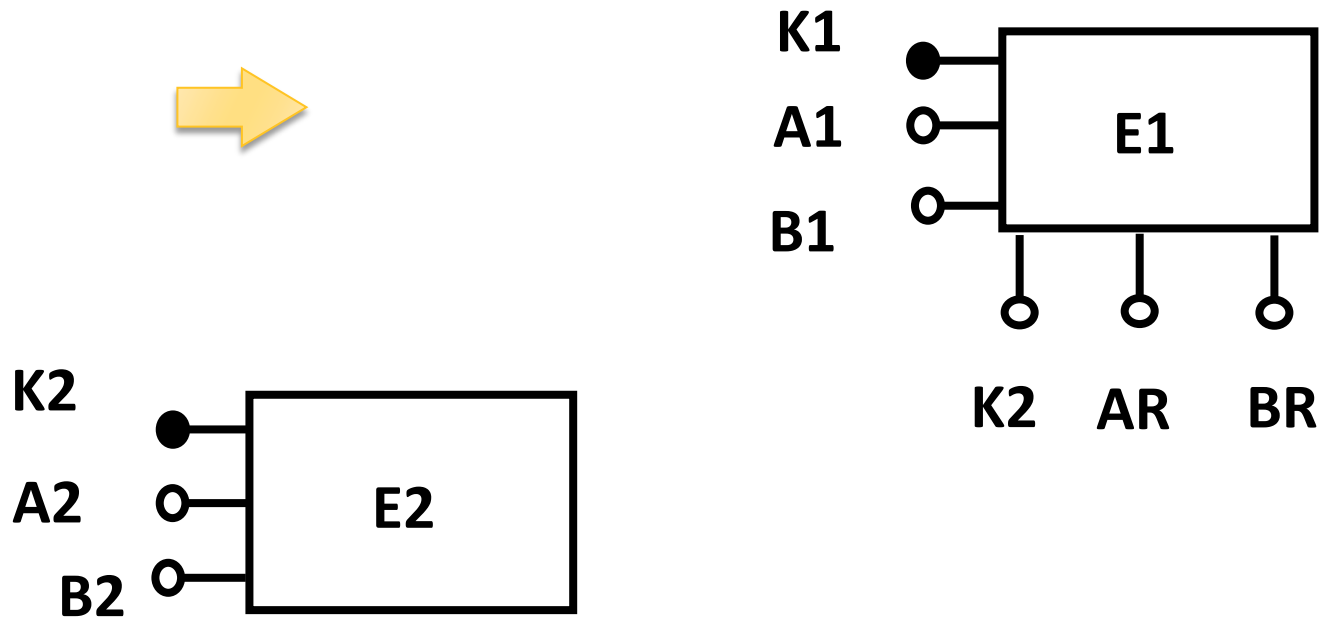
$E1(\underline{K1}, A1, B1, K2, AR, BR)$

$E2(\underline{K2}, A2, B2)$

- ▶ Se E1 partecipa con cardinalità (0,1) la soluzione a due relazioni ha valori nulli in K2, AR, BR per le istanze di E1 che non partecipano all'associazione

Associazione binaria 1 a N

► equivale a:



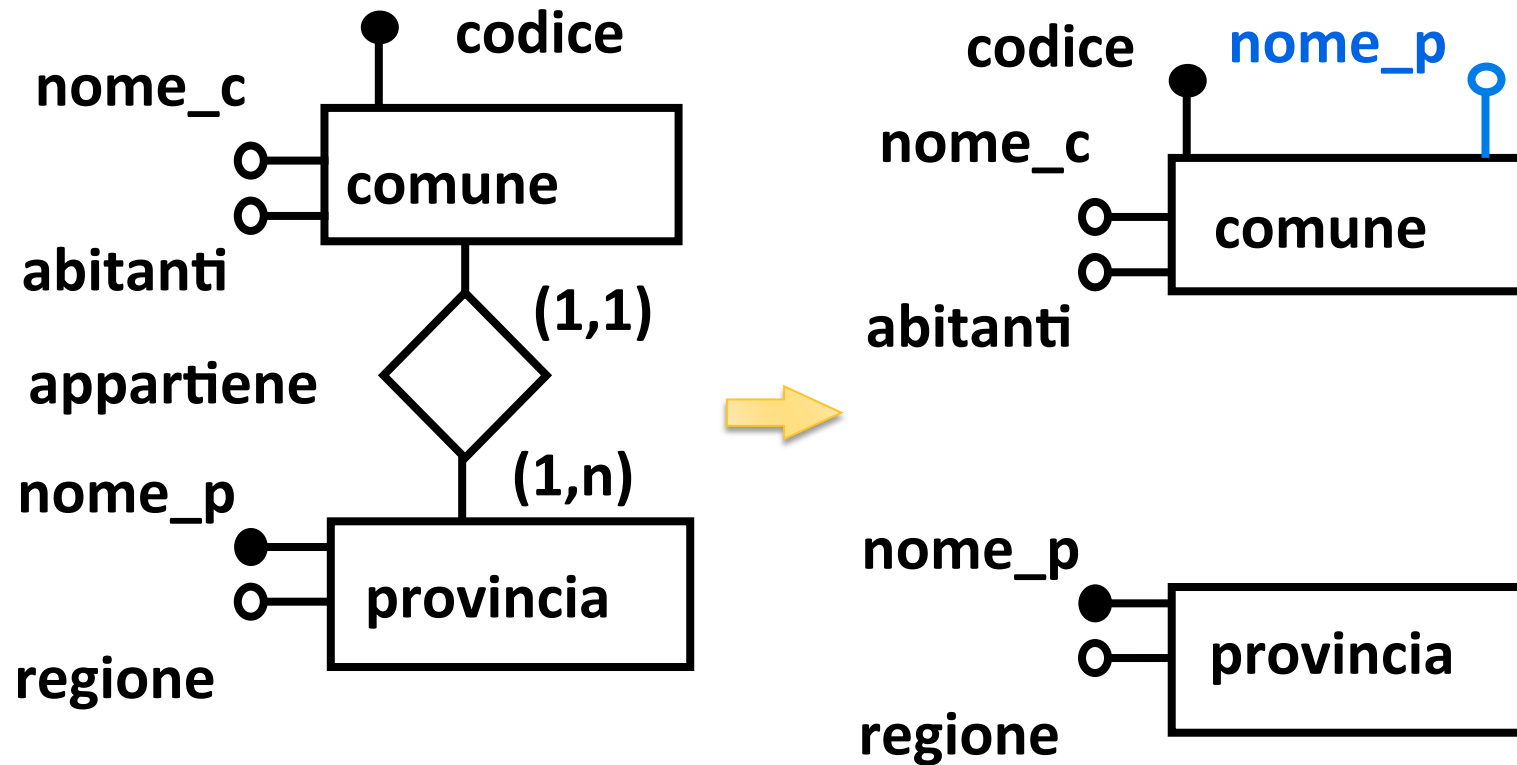
Associazione binaria 1 a N

- ▶ **Attenzione** : in questo caso, poiché la partecipazione di E1 è 0,1 o 1,1, si nota facilmente che **ad un dato valore di K1 corrisponde uno e un sol valore di K2** (non è vero il contrario), quindi si può dire che **K1 implica K2** o, anche, che esiste una **dipendenza funzionale da K1 a K2**
- ▶ nella soluzione a 3 relazioni la chiave della relazione che traduce l'associazione è **riducibile a K1**:

$E1(\underline{K1}, A1, B1)$, $E2(\underline{K2}, A2, B2)$

$R(\underline{K1}, K2, AR, BR)$

Associazione binaria 1 a N es.



(senza attributi sull'associazione)

Associazione binaria 1 a N es.

PROVINCIA (NOME_P, REGIONE, ...)

COMUNE (CODICE, NOME_C, ABITANTI, NOME_P, ...)

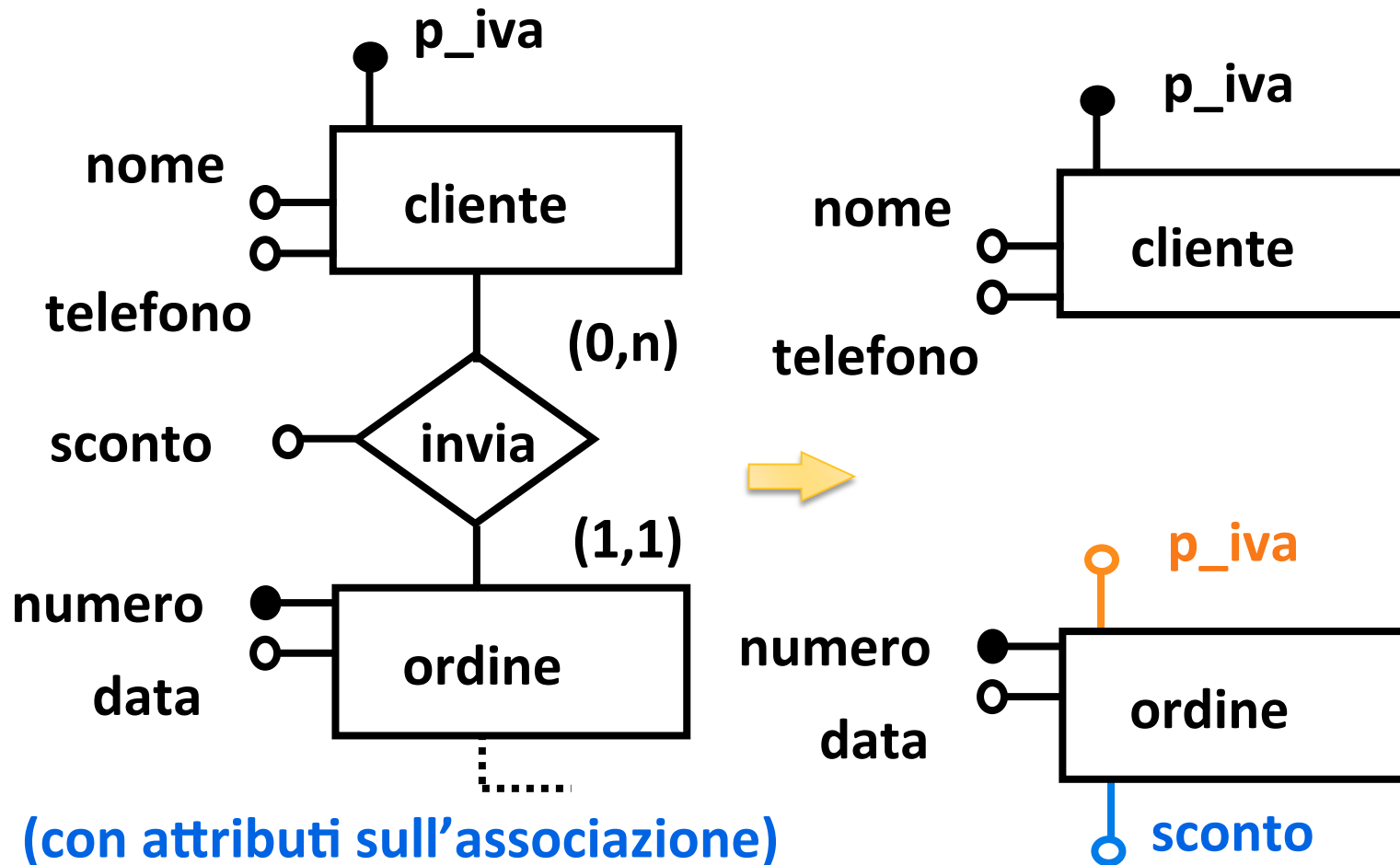
FK: NOME_P REFERENCES PROVINCIA

Associazione binaria 1 a N es.

```
CREATE TABLE PROVINCIA
(NOME_P ... NOT NULL,
 REGIONE ... PRIMARY KEY (NOME_P) );
```

```
CREATE TABLE COMUNE
(CODICE ... NOT NULL, NOME_C ...
 ABITANTI ..., NOME_P ... NOT NULL
 PRIMARY KEY (CODICE)
 FOREIGN KEY NOME_P
 REFERENCES PROVINCIA);
```

Associazione binaria 1 a N es.



Associazione binaria 1 a N es.

traduzione con due relazioni:

CLIENTE (P_IVA, NOME, TELEFONO)

ORDINE (NUMERO, DATA, P_IVA, SCONTO)

FK: P_IVA REFERENCES CLIENTE

Associazione binaria 1 a N es.

```
CREATE TABLE CLIENTE (P_IVA ... NOT NULL,  
NOME ..., TELEFONO ..., PRIMARY KEY (P_IVA));
```

```
CREATE TABLE ORDINE (NUMERO ... NOT NULL,  
DATA ... P_IVA ... NOT NULL, SCONTO ...,  
PRIMARY KEY (NUMERO)  
FOREIGN KEY P_IVA REFERENCES CLIENTE);
```

con tre relazioni:



Associazione binaria 1 a N es.

CLIENTE (P_IVA, NOME, TELEFONO)

ORDINE (NUMERO, DATA)

INVIA (NUMERO, P_IVA, SCONTO)

FK: P_IVA REFERENCES CLIENTE

FK: NUMERO REFERENCES ORDINE

Associazione binaria 1 a N es.

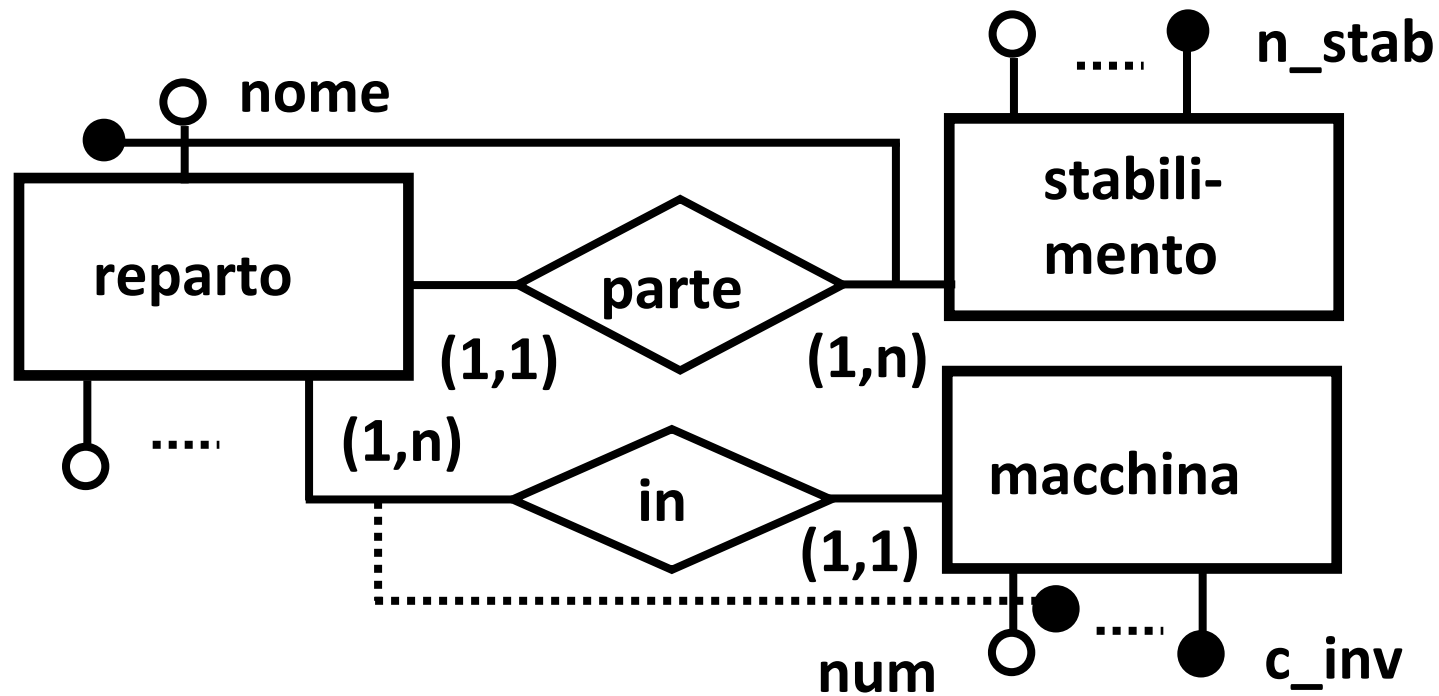
```
CREATE TABLE CLIENTE (P_IVA ... NOT NULL,  
    NOME ... ,TELEFONO ... , PRIMARY KEY (P_IVA)) ;
```

```
CREATE TABLE ORDINE (NUMERO ... NOT NULL,  
    DATA ... PRIMARY KEY (NUMERO)) ;
```

```
CREATE TABLE INVIA  
(P_IVA ... NOT NULL, NUMERO ... NOT NULL,  
    SCONTO ... , PRIMARY KEY (NUMERO)  
FOREIGN KEY P_IVA REFERENCES CLIENTE  
FOREIGN KEY NUMERO REFERENCES  
ORDINE) ;
```


Associazione binaria 1 a N es.

Con identificazione esterna



Associazione binaria 1 a N es.

STABILIMENTO (N_STAB, ...)

REPARTO (NOME, N_STAB, ...)

FK: N_STAB REFERENCES STABILIMENTO

MACCHINA (NUM, NOME, N_STAB, ...)

FK: NOME REFERENCES REPARTO

FK: N_STAB REFERENCES STABILIMENTO

AK: C_INV

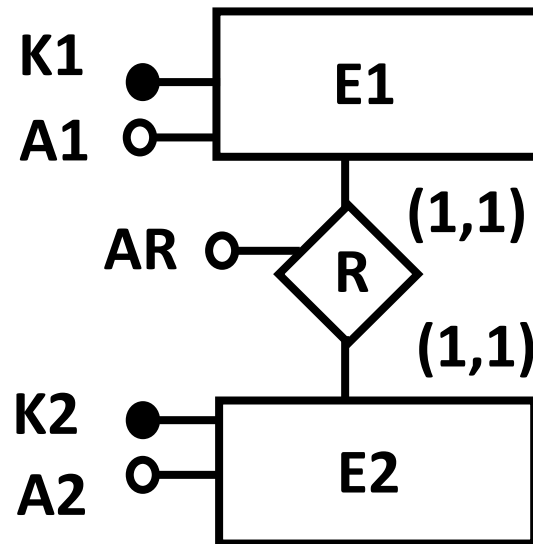
Associazione binaria 1 a N es.

```
CREATE TABLE STABILIMENTO (N_STAB ... NOT  
NULL, ..., ..., PRIMARY KEY (N_STAB));
```

```
CREATE TABLE REPARTO (NOME ... NOT NULL,  
N_STAB ... NOT NULL... PRIMARY KEY (NOME,  
N_STAB) FOREIGN KEY N_STAB REFERENCES  
STABILIMENTO
```

```
CREATE TABLE MACCHINA (NUM ... NOT NULL,  
NOME ... NOT NULL, N_STAB ... NOT NULL, ...,  
PRIMARY KEY (NUM, NOME, N_STAB )  
FOREIGN KEY NOME REFERENCES REPARTO  
FOREIGN KEY N_STAB REFERENCES STABILIMENTO);
```

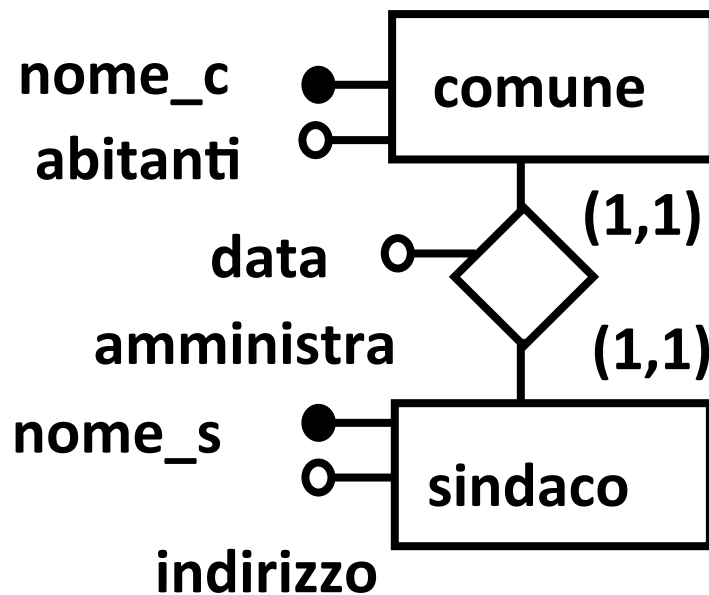
Associazione binaria 1 a 1



► traduzione con una relazione:

E12 (K1, A1, K2, A2, AR)

Associazione binaria 1 a 1



► traduzione con una relazione:



Associazione binaria 1 a 1

AMMINISTRAZIONE (NOME_C, ABITANTI, NOME_S,
INDIRIZZO, DATA)

AK: NOME_S

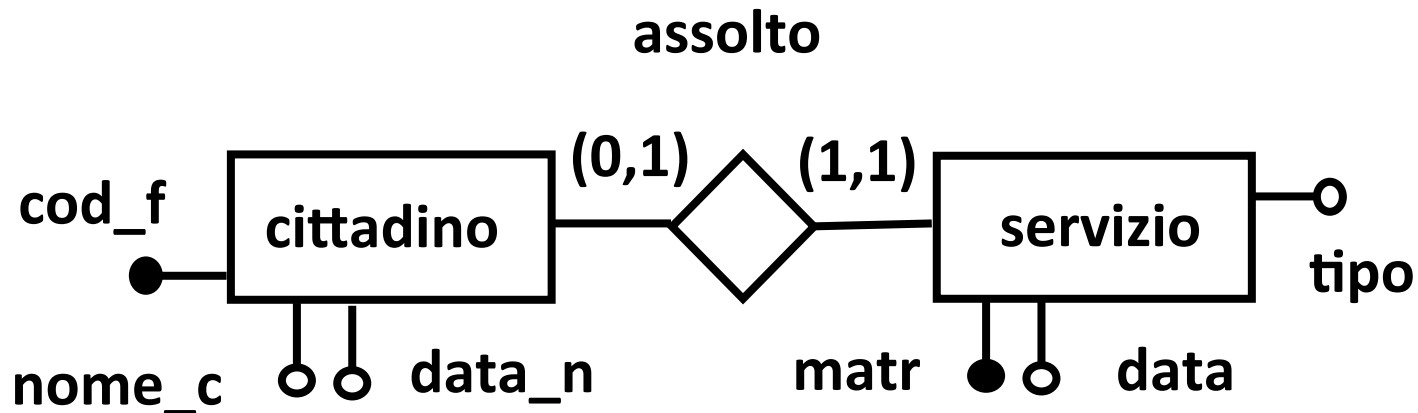
```
CREATE TABLE AMMINISTRAZIONE
(NOME_C    ... NOT NULL,    ABITANTI    ... ,
 NOME_S    ... NOT NULL UNIQUE,
 INDIRIZZO ... ,    DATA    ... ,
PRIMARY KEY (NOME_C) );
```

se le cardinalità minime sono entrambe 1 la
chiave può essere indifferentemente K1 o K2
si sceglierà quella più significativa

Associazione binaria 1 a 1

- se la cardinalità di E2 è 0,1 e quella di E1 è 1,1 allora la **chiave sarà K2** ; E2 è l'entità con maggior numero di istanze alcune della quali non si associano, ci saranno quindi valori nulli in corrispondenza di K1, K1 in questo caso non potrebbe essere scelta
- se la cardinalità è 0,1 da entrambe le parti allora le relazioni saranno due per **l'impossibilità di assegnare la chiave** all'unica relazione a causa della presenza di valori nulli sia su K1 che su K2

Associazione binaria 1 a 1



CITTADINO (COD_F, NOME_C, INDIRIZZO,
DATA_N, **MATR**, **DATA**, TIPO)

```
CREATE TABLE CITTADINO
(COD_F ... NOT NULL, NOME_C ... NOT NULL,
INDIRIZZO ..., DATA_N ..., MATR ..., DATA...,
TIPO ..., PRIMARY KEY (COD_F));
```


Associazione binaria 1 a 1

- ▶ Traduzione con due relazioni
 - ▶ l'associazione può essere **compattata** con l'entità che partecipa **obbligatoriamente** (una delle due se la partecipazione è obbligatoria per entrambe) la discussione sulla chiave è analoga al caso di traduzione con una relazione

E1 (K1, A1, B1,...)

E2 (K2, A2, B2,... K1, AR, BR)

Associazione binaria 1 a 1

- ▶ Traduzione con tre relazioni
 - ▶ la chiave della relazione che traduce l'associazione può essere indifferentemente K1 o K2, non ci sono problemi di valori nulli

E1 (K1, A1, B1,...)

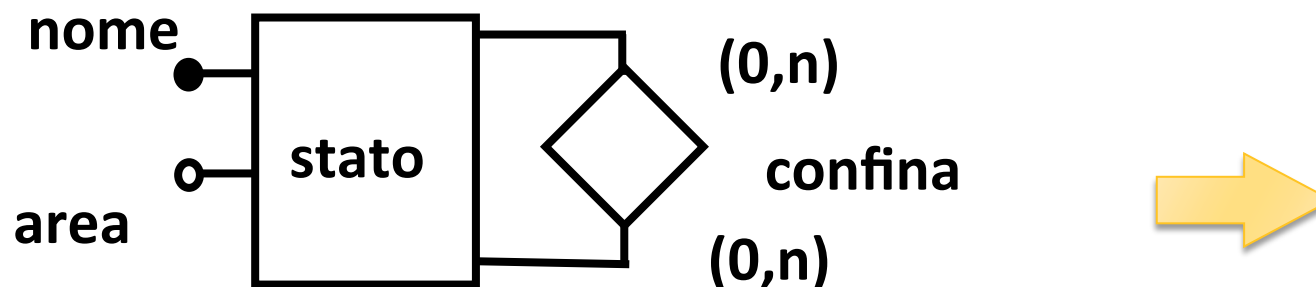
E2 (K2, A2, B2,...)

R (K1, K2, AR, BR,...)

Auto associazione N a M

viene tradotta con:

- ▶ una relazione per l'entità ed
- ▶ una per l'associazione,
 - quest'ultima contiene due volte la chiave dell'entità, è necessario, però modificare i nomi degli attributi, per non avere **omonimia**



Auto associazione N a M

STATO(NOME, AREA)

CONFINA(STATO_A, STATO_B)

FK: STATO_A REFERENCES STATO
STATO_B REFERENCES STATO

CREATE TABLE STATO

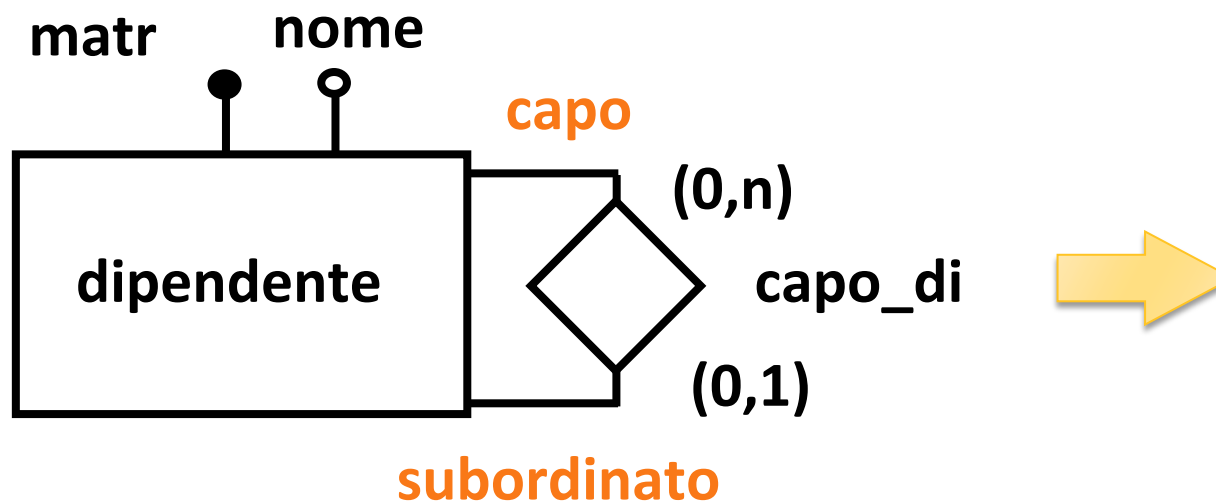
(**NOME** ... NOT NULL, AREA ...
PRIMARY KEY (NOME));

CREATE TABLE CONFINA

STATO_A ... NOT NULL, **STATO_B** ... NOT NULL,
PRIMARY KEY (**STATO_A**, **STATO_B**)
FOREIGN KEY (STATO_A)
REFERENCES STATO
FOREIGN KEY (STATO_B)
REFERENCES STATO);

Auto associazione 1 a N

- ▶ è traducibile con **una sola relazione** che contiene due volte l'attributo chiave: una volta come **chiave** ed una come **riferimento** all'istanza connessa, con nome diverso per specificare il **ruolo**



Auto associazione 1 a N

DIPENDENTE (MATR, NOME, CAPO)

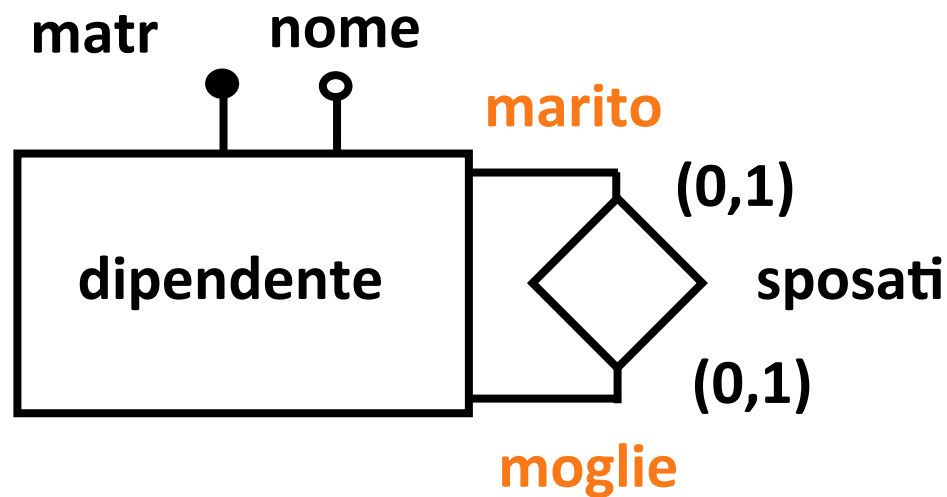
FK: CAPO REFERENCES DIPENDENTE

```
CREATE TABLE DIPENDENTE
(MATR ... NOT NULL, NOME ..., CAPO ...
PRIMARY KEY (MATR)
FOREIGN KEY (CAPO)
REFERENCES DIPENDENTE) ;
```

- ▶ nel caso di associazione **1 ad 1** il concetto di **ruolo** assume maggiore importanza:



Auto associazione 1 a 1



- ▶ su **entrambi rami** è bene specificare il **ruolo**: conviene la soluzione con due relazioni per evitare ridondanze, vincoli ed eccesso di valori nulli.

Auto associazione 1 a 1

DIPENDENTE (MATR, NOME)

SPOSATI (MOGLIE, MARITO)

FK: **MOGLIE REFERENCES** DIPENDENTE

FK: **MARITO REFERENCES** DIPENDENTE

Auto associazione 1 a 1

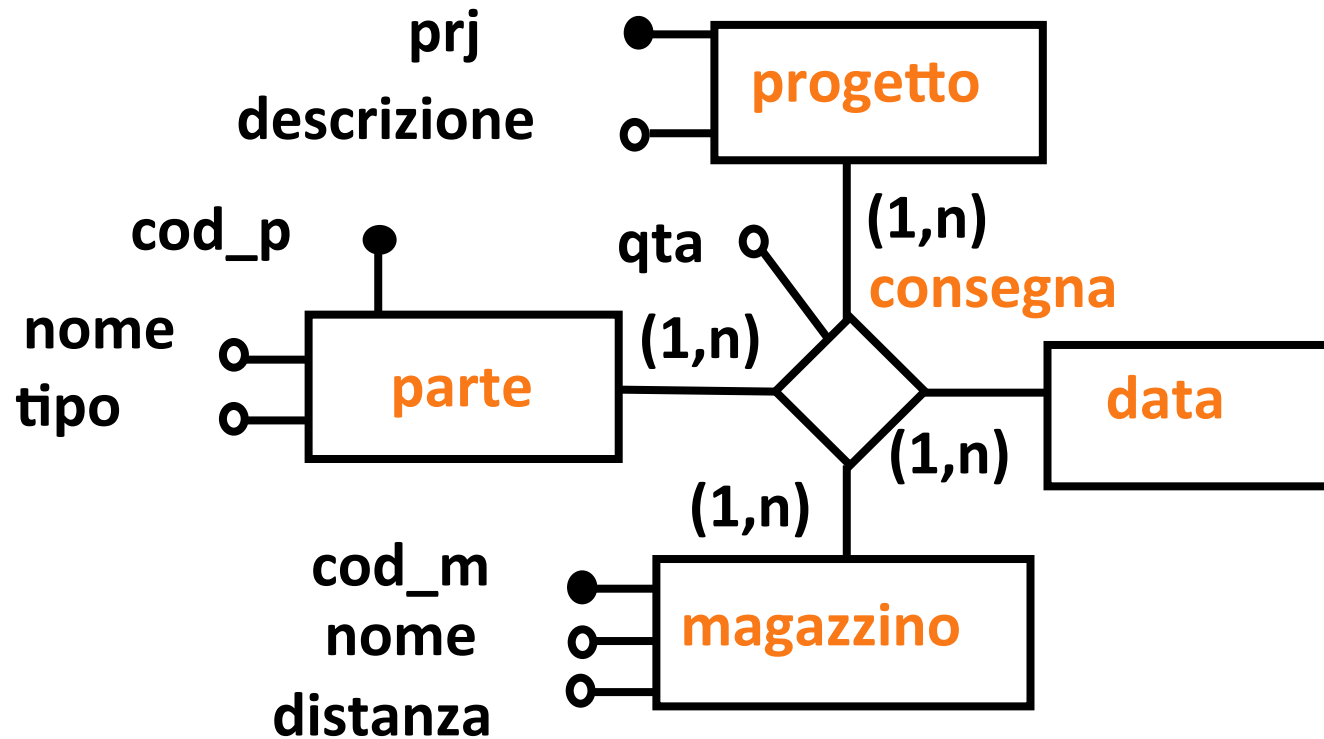
```
CREATE TABLE DIPENDENTE (MATR ... NOT  
NULL, NOME ..., PRIMARY KEY (MATR)
```

```
CREATE TABLE SPOSATI  
(MOGLIE ... NOT NULL, MARITO ... NOT NULL  
PRIMARY KEY (MOGLIE)  
FOREIGN KEY (MOGLIE) REFERENCES DIPENDENTE  
FOREIGN KEY (MARITO) REFERENCES DIPENDENTE) ;
```

Associazione n-aria

- ▶ segue la traduzione standard
- ▶ talvolta, nella relazione che traduce l'associazione, la chiave ottenuta componendo le chiavi di tutte le entità partecipanti è una **superchiave**, cioè una chiave composta il cui set di componenti **non è minimale** (la chiave vera è un sottoinsieme)
- ▶ **Esempio: progetti-parti-magazzini**

Associazione n-aria



Associazione n-aria

PROGETTO (PRJ, DESCRIZIONE)

PARTE (COD_P, NOME, TIPO)

MAGAZZINO (COD_M, NOME, DISTANZA)

```
CREATE TABLE PROGETTO (PRJ... NOT NULL,  
    DESCRIZIONE... , PRIMARY KEY (PRJ));
```

```
CREATE TABLE PARTE (COD_P ... NOT NULL,  
    NOME..., TIPO..., PRIMARY KEY (COD_P));
```

```
CREATE TABLE MAGAZZINO (COD_M... NOT NULL,  
    NOME ... , DISTANZA..., PRIMARY KEY (COD_M));
```

non c'è una relazione per la data

la data era un'entità fittizia messa nello schema per garantire l'unicità delle consegne, comparirà infatti nella definizione della chiave

Associazione n-aria

l'associazione diventa:

CONSEGNA (PRJ, COD_P, COD_M, DATA, QTA)

FK: PRJ REFERENCES PROGETTO

FK: COD_M REFERENCES MAGAZZINO

FK: COD_P REFERENCES PARTE

```
CREATE TABLE CONSEGNA (PRJ ... NOT NULL,  
  COD_P... NOT NULL, COD_M... NOT NULL,  
  DATA... NOT NULL, QTA ...  
  PRIMARY KEY (PRJ, COD_P, COD_M, DATA)  
  FOREIGN KEY (PRJ) REFERENCES PROGETTO  
  FOREIGN KEY (COD_M) REFERENCES MAGAZZINO  
  FOREIGN KEY (COD_P) REFERENCES PARTE);
```

ipotizziamo che (PRJ, COD_P, COD_M, DATA)
sia una superchiave:



Associazione n-aria

- ▶ una parte esiste in un solo magazzino, quindi **COD_P** è associato ad un solo **COD_M**, cioè determina COD_M, allora la presenza di COD_M nella chiave è **ridondante**:

CONSEGNA (PRJ, COD_P, COD_M, DATA, QTA)

FK: PRJ REFERENCES PROGETTO

FK: COD_M REFERENCES MAGAZZINO

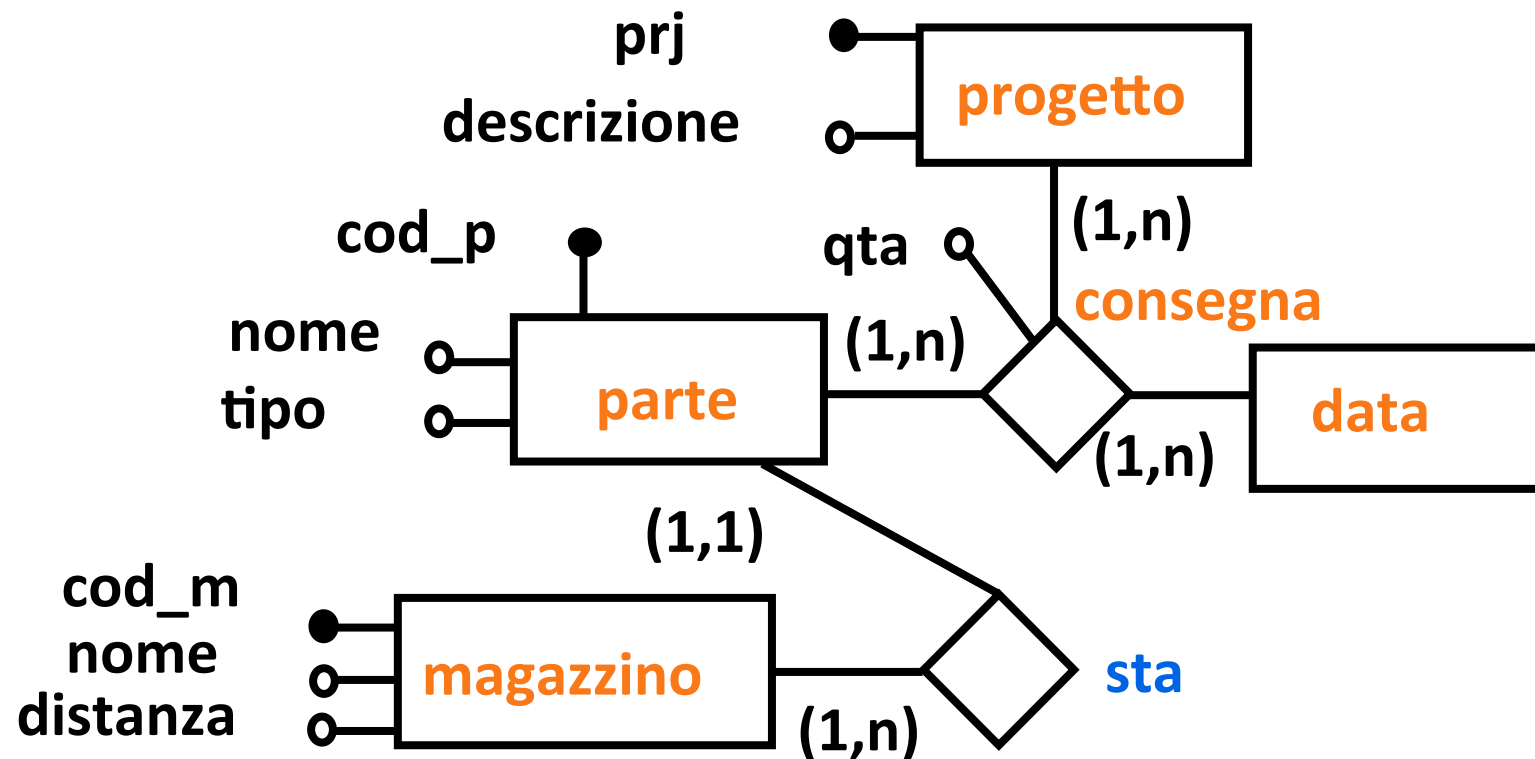
FK: COD_P REFERENCES PARTE

- **COD_M è ridondante anche nella relazione**



Associazione n-aria

- ▶ infatti, se una parte esiste in un solo magazzino:



Associazione n-aria

PROGETTO (PRJ, DESCRIZIONE)

MAGAZZINO (COD_M, NOME, DISTANZA)

PARTE (COD_P, NOME, TIPO, COD_M)

FK: COD_M REFERENCES MAGAZZINO

Associazione n-aria

```
CREATE TABLE PROGETTO (PRJ... NOT NULL,  
    DESCRIZIONE... , PRIMARY KEY (PRJ)) ;
```

```
CREATE TABLE MAGAZZINO (COD_M... NOT NULL,  
    NOME ... , DISTANZA ... , PRIMARY KEY (COD_M)) ;
```

```
CREATE TABLE PARTE (COD_P ... NOT NULL,  
    NOME... , TIPO... , COD_M... NOT NULL,  
    PRIMARY KEY (COD_P) , FOREIGN KEY (COD_M)  
    REFERENCES MAGAZZINO) ;
```

Associazione n-aria

- la chiave forestiera nella relazione PARTE traduce l'associazione STA (1 a N) ed elimina le ripetizioni nella relazione CONSEGNA
- l'associazione diventa:

CONSEGNA (PRJ, COD_P, DATA, QTA)

FK: PRJ REFERENCES PROGETTO

FK: COD_P REFERENCES PARTE

```
CREATE TABLE CONSEGNA (PRJ ... NOT NULL,  
                        COD_P... NOT NULL, DATA... NOT NULL, QTA  
PRIMARY KEY (PRJ, COD_P, DATA)  
FOREIGN KEY (PRJ) REFERENCES PROGETTO  
FOREIGN KEY (COD_P) REFERENCES PARTE);
```

Commento

- ▶ nel caso precedente la **dipendenza** tra **magazzino e parte** non era stata espressa sulla associazione n-aria, abbiamo ipotizzato di scoprirla nella fase di progetto logico
- ▶ **se il progetto concettuale è ben fatto casi del genere non sono frequenti**
- ▶ diverso è il caso in cui si vogliono esprimere dei vincoli che richiederebbero un uso complicato di entità di collegamento con identificazione esterna
- ▶ **il ricontrollo delle chiavi delle relazioni è quindi importante**