Tecnologie dell'Informazione e della Comunicazione per la Protezione Civile
Sistemi Informativi e Basi di Dati

Gianpaolo Cugola
Politecnico di Milano – Dip. di Elettronica e Informazione
http://www.elet.polimi.it/people/cugola

Sommario e testi

- Sommario
  - I sistemi informativi
  - Le basi di dati e i DBMS
    - Progettazione logica: il modello relazionale
    - Interrogazione: il linguaggio SQL
    - Progettazione concettuale: il modello Entità-Relazioni
- Testo di riferimento
  - Atzeni, Ceri, Paraboschi, Torlone: "Basi di Dati", McGraw-Hill, 1999

I sistemi informativi

- Nello svolgimento di ogni attività sono essenziali la disponibilità di informazioni...
- ... e la capacità di gestirle in maniera efficace
- Ogni organizzazione è dotata di un *sistema informativo* che organizza e gestisce le informazioni utili all'organizzazione
- Il concetto di sistema informativo non è necessariamente legato all'automazione e alla disponibilità di sistemi informatici
  - Gli archivi delle banche e dei servizi anagrafici esistono da secoli

Informazioni vs. dati

- Con il termine *dati* indichiamo quei concetti che una volta interpretati e correlati opportunamente forniscono *informazioni* su un certo dominio
- Esempio: il *dato* `Mario Rossi 25775`
  - Di per se non porta grande informazione
  - Ma se inviato in risposta alla domanda: "chi è il responsabile dell'ufficio tecnico e qual'è la sua matricola" può essere interpretato e diventa *informazione*

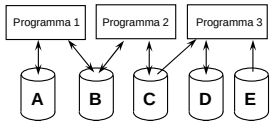
Base di dati

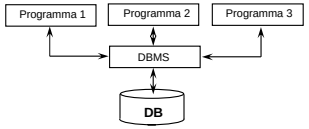
- Una *base di dati* è una collezione di dati utilizzati per rappresentare le informazioni di interesse per un sistema informativo
- Dall'avvento del calcolatore quest'ultimo è diventato lo strumento principe per la conservazione e la manipolazione delle basi di dati di competenza dei diversi sistemi informativi

I sistemi per la gestione di basi di dati

- Nascono per centralizzare la gestione dei dati, affidandola ad un unico programma: il DBMS (data base management system, o sistema per la gestione di basi di dati)

Ieri: diversi archivi su memoria di massa direttamente usati da programmi diversi





Oggi: unico DB gestito da un singolo DBMS che centralizza l'acceso ai dati da parte di diversi programmi

Benefici derivanti dall'uso dei DBMS

- Se ben usati, i DBMS forniscono i seguenti vantaggi:
  - Unica rappresentazione dei dati
    - Meno possibilità di ridondanza e inconsistenza
  - Accesso disciplinato attraverso il DBMS
    - Privatezza, ...
  - Gestione della memoria di massa
    - All'utente viene presentata una visione logica dei dati, nascondendo i dettagli relativi ai meccanismi di salvataggio su memoria di massa, con conseguente portabilità della base di dati
  - Gestione degli accessi concorrenti
    - L'accesso ad un dato è negato se c'è già un programma che sta manipolando quel dato

DB vs. DBMS

- Con il termine Data Base (DB) indichiamo l'insieme dei dati mantenuti in un DBMS
- Con il termine Data Base Management System (DBMS) indichiamo un software il cui obiettivo sia la gestione di data base

I modelli dei dati

- Il modello dei dati indica
  - Come sono organizzate, strutturate e presentate le informazioni agli utenti del DBMS
  - Quali operazioni sono disponibili sui dati
- Esistono diversi modelli dei dati effettivamente usati nei DBMS:
  - Gerarchico (metà anni sessanta)
  - Reticolare (Codasyl - 1973, 1978)
  - Relazionale (inizio anni '80)
  - Object-oriented (inizio anni '90)

Schema e istanza della base di dati

- Lo schema di un DB è la descrizione dei dati e della loro struttura nel DB, definito attraverso l'attività di progettazione del DB
  - Esempio:
    - Una singola tabella con una riga per ogni record e una colonna per ogni campo
    - Un insieme di tabelle logicamente collegate da campi univoci comuni (chiavi)
    - Un insieme di "oggetti" collegati da riferimenti simbolici
- Il DB contiene istanze (o occorrenze) dei dati

Il modello relazionale

- Oggi è il più diffuso modello dei dati
  - Grazie alla sua semplicità, eleganza e flessibilità
- Un data base è organizzato in un insieme di relazioni (o tabelle)
- Ciascuna tabella è un insieme di tuple (record)
- Ciascuna tupla è una sequenza di attributi (campi)
- L'attributo è l'unità elementare di informazione, contraddistinto dal dominio, cioè dall'insieme predefinito di valori che può assumere

Esempio di tabella

Conto_corrente

Num_CC	Nome	Indirizzo	Saldo
1	Rossi	V.Roma, 13	5.876.430
2	Bianchi	V. Verdi, 2	2.654.390
3	Ferrari	P.za Po,3	1.110.110

Gli attributi sono:
- Num_CC, con dominio intero
- Nome, con dominio stringa
- Indirizzo, con dominio stringa
- Saldo, con dominio intero

- Il *grado della relazione* (numero di attributi) è 4
- La *cardinalità della relazione* (numero di tuple) è 3

Lo schema di un DB relazionale

- È dato dall'insieme degli schemi delle tabelle appartenenti al DB
- Lo schema di una tabella è semplicemente l'elenco degli attributi, ciascuno col suo tipo (o dominio)
  - Ad esempio:

```
Relation Conto_corrente (Num_CC: integer,
  Nome: char(20), Indirizzo: char(20),
  Saldo: integer)
Relation Movimento (Num_CC: integer,
  Data_mov: date, Num_mov: integer,
  Importo: integer, Causale: char(1))
```

- Oltre ai soliti tipi, nei DB si trovano spesso i tipi date, time e money

Considerazioni sul modello relazionale

- Siamo vincolati ad introdurre informazioni che soddisfino lo schema
  - Ad es. nella relazione Conto_corrente possiamo introdurre solo il nome e l'indirizzo del cliente, non il suo numero di telefono o il codice fiscale. Per inserire queste informazioni dovremmo prima modificare lo schema
- Partendo dalle relazioni esistenti è possibile ricavare informazioni non direttamente disponibili nel DB, ad es. l'elenco dei clienti che abitano in una certa zona e hanno versato più di un milione nel 1995
- Questo tipo di operazioni viene fatto attraverso il *query language* (o linguaggio di interrogazione del DBMS)

Accesso alle tuple

- L'accesso ad una certa tupla (o ad un insieme di tuple) di una relazione è sempre ed esclusivamente di tipo associativo (non posizionale): avviene in base al valore contenuto nella tupla
- Ovvero, posso chiedere al DBMS il saldo di Rossi
  - Trova la tupla in cui l'attributo Nome vale "Rossi" e leggi l'attributo Saldo
- Non posso chiedere il valore dell'attributo Nome della terza tupla
  - Le tuple non sono ordinate e non c'è modo di fare un accesso diretto

Chiavi delle relazioni

- A causa dell'accesso associativo, è importante dotare le relazioni di una "chiave"
- *Una chiave è un insieme minimo di attributi il cui valore identifica univocamente una tupla*
- Serve per poter accedere ad una singola tupla
  - Ad es. nel caso del Conto_corrente si vuole poter aggiornare il Saldo di una specifica tupla, corrispondente ad un ben preciso conto corrente. La chiave sarà pertanto l'attributo Num_CC
  - Nel caso del Movimento, la chiave sarà data dall'insieme degli attributi Num_CC, Data_mov, Num_mov

Chiavi delle relazioni

- C è una chiave per una relazione R se valgono le seguenti proprietà:
  - *Univocità*: non possono esistere due tuple di R con lo stesso valore di C
  - *Minimalità*: eliminando un attributo da C la proprietà precedente decade
- Per una stessa relazione possono esistere più *chiavi candidate*
  - La *chiave primaria* della relazione sarà una tra queste
  - Spesso sulla chiave primaria non si accettano valori nulli

Le operazioni relazionali

- Ci sono operazioni che servono per combinare relazioni, limitandosi a leggere il contenuto del DB. Restituiscono sempre una relazione
  - Operazioni unarie
    - hanno come operando un'unica relazione
  - Operazioni binarie
    - hanno come operando due relazioni
  - Operazioni insiemistiche
    - corrispondono alle solite operazioni di unione, differenza e intersezione
- Ci sono operazioni che servono a modificare il contenuto del DB
  - Aggiunta e riomozione di record
- Ci sono operazioni che servono a modificare lo schema del DB

Le operazioni unarie - selezione

- La selezione restituisce una relazione che è strutturalmente identica all'operando (ha lo stesso schema), ma contiene un sottoinsieme delle tuple dell'operando
- Fa una selezione delle tuple della relazione "operando", utilizzando un criterio di selezione:
  - Il criterio di selezione è una espressione logica (*predicato*) che viene valutata (vero/falso) su ciascuna tupla: le tuple per cui l'espressione vale "vero" sono selezionate e fanno parte del risultato, le altre, per cui l'espressione vale "falso" sono scartate.

Esempio di selezione

- Selezioniamo dalla relazione Conto_corrente le tuple in cui Saldo > 2.000.000

Conto_corrente

Num_CC	Nome	Indirizzo	Saldo
1	Rossi	V.Roma, 13	5.876.430
2	Bianchi	V. Verdi, 2	2.654.390
3	Ferrari	P.za Po,3	1.110.110

Conto_corrente

Num_CC	Nome	Indirizzo	Saldo
1	Rossi	V.Roma, 13	5.876.430
2	Bianchi	V. Verdi, 2	2.654.390

Le operazioni unarie - proiezione

- Mentre la selezione elimina delle righe della tabella operando, la proiezione elimina delle colonne
- Contrariamente al caso della selezione il criterio di eliminazione non dipende da un'espressione da valutare, ma si dà direttamente l'insieme degli attributi che vanno mantenuti

Esempio di proiezione

- Proiettiamo la relazione Movimento sugli attributi Data_mov e Importo

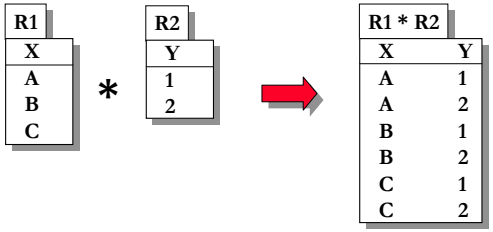
Movimento

Num_CC	Data_mov	Num_mov	Importo	Causale
1	1-1-96	1	100.000	P
1	4-2-96	1	300.000	A
2	3-1-96	1	200.000	P
3	5-5-96	1	100.000	P
4	1-1-96	1	100.000	S
4	6-9-96	1	100.000	A

- Nota: eliminando delle colonne può darsi che tra le tuple così ottenute esistano dei duplicati. Poiché le relazioni sono insiemi, i duplicati non sono ammessi, e quindi vengono eliminati
  - Ad es. il movimento del CC 1 del 1-1-96 è un duplicato e viene eliminato

Le operazioni binarie - prodotto cartesiano

- Il prodotto cartesiano crea una relazione avente per tuple tutte le possibili combinazioni ottenibili giustapponendo una tupla del primo operando con una tupla del secondo
- Esempio:



Le operazioni binarie - join naturale

- Il prodotto cartesiano raramente è utile, perché generalmente si vogliono ottenere solo le combinazioni di tuple tra le quali esiste una certa corrispondenza, o vale una certa proprietà
- Join naturale: si combinano tra loro solo le tuple in cui valori delle relazioni in due attributi aventi dominio uguale (e nome uguale) verificano la proprietà di uguaglianza
- Nella tabella risultante, si considera la colonna uguale una volta sola

Le operazioni binarie - join naturale: Esempio

Impiegato	Reparto
Rossi	vendite
Neri	produzione
Verdi	produzione

Reparto	Dirigente
produzione	Mori
produzione	Bianchi
vendite	Bruni

Impiegato	Reparto	Dirigente
Rossi	vendite	Bruni
Neri	produzione	Mori
Neri	produzione	Bianchi
Verdi	produzione	Mori
Verdi	produzione	Bianchi

Il linguaggio SQL

- I DBMS relazionali dispongono di un linguaggio standard per le interrogazioni, quindi anche per fare selezioni e proiezioni
- Questo linguaggio si chiama SQL (structured query language)
- Il costrutto più comune è il blocco SELECT:

```
SELECT lista di attributi
FROM relazione
WHERE predicato
```


Esempi di operazioni SQL

- La selezione riportata nell'esempio precedente si scrive:

```
SELECT *
FROM Conto_corrente
WHERE Saldo > 2.000.000
```

- La proiezione dell'esempio precedente si scrive:

```
SELECT Data_mov, Importo
FROM Movimento
```

- Nota: nei DBMS commerciali i duplicati sono ammessi. Per eliminarli occorre indicarlo esplicitamente, attraverso la parola chiave DISTINCT:

```
SELECT DISTINCT Data_mov, Importo
FROM Movimento
```


Operazioni combinate in SQL

- Una singola espressione SQL può indicare selezione e proiezione insieme:

```
SELECT Num_CC, Importo
FROM Movimento
WHERE Data_mov = 1-1-96
```

- L'operazione descritta seleziona le tuple in cui l'attributo Data_mov ha il valore prescritto, poi sopprime gli attributi diversi da Num_CC e Importo

Join in SQL

- Voglio sapere i nomi dei correntisti interessati da ciascun movimento: devo combinare le tuple di Conto_corrente con le tuple di Movimento aventi uguale Num_CC
- In SQL scrivo:

```
SELECT *
FROM Conto_corrente, Movimento
WHERE  Conto_corrente.Num_CC =
       Movimento.Num_CC
```


Join, selezione e proiezione insieme

- Il join è combinabile con la selezione e la proiezione in un'unica operazione SQL: semplicemente il predicato non indicherà solo la corrispondenza tra tuple, ma anche un criterio cui le tuple del risultato dovranno essere conformi
- Voglio conoscere importo e causale dei movimenti di Rossi del 1-1-96:

```
SELECT Importo, Causale
FROM Conto_corrente, Movimento
WHERE     Conto_corrente.Num_CC =
          Movimento.Num_CC AND
          Nome = "Rossi" AND
          Data_mov = 1-1-96
```


Le operazioni insiemistiche

- Corrispondono alle normali operazioni sugli insiemi
- Occorre notare che per avere come risultato dei veri insiemi occorre sempre indicarlo esplicitamente usando la keyword DISTINCT
  - Altrimenti ad es. l'unione darà semplicemente un risultato contenente le tuple del primo operando e quelle del secondo, duplicati compresi
- In SQL queste operazioni si chiamano rispettivamente
  - UNION
  - MINUS
  - INTERSECT

Esempio di interrogazione

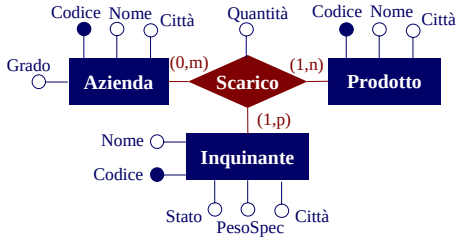
- Estraiamo i CC che hanno un saldo maggiore di 2.000.000 e per i quali non è stato fatto alcun movimento per un importo maggiore di 1.000.000

```
SELECT Num_CC
FROM     Conto_corrente
WHERE    Saldo > 2.000.000
MINUS
SELECT Num_CC
FROM     Movimento
WHERE    Importo > 1.000.000
```

Stili di interrogazione: imperativo e dichiarativo

- L'SQL supporta uno stile di interrogazione dichiarativo, nel senso che le query specificano le caratteristiche del risultato ma non indicano come estrarlo
  - Ad es. nelle selezioni non diciamo se la tabella deve essere scandita dall'alto o dal basso, o quale parte della condizione deve essere valutata per prima
- Le query vengono interpretate dal DBMS, che provvede anche a ottimizzarle
  - Ad es. dovendo fare un join e una selezione conviene fare la selezione prima, per avere meno tuple su cui valutare la condizione del join

Esempio di riferimento: schemi concettuale e logico




```
AZIENDA(CodA, NomeA, Grado, Città)
INQUINANTE(CodI, NomeI, Stato, PesoSpec, Città)
PRODOTTO(CodP,NomeP, Città)
SCARICO(CodA, CodI, CodP, Quantità)
```


Esempio di riferimento: istanza della base di dati

AZIENDA

CodA	NomeA	Grado	Città
A1	Colle & Vernici	20	Torino
A2	Factotum SpA	10	Milano
A3	SuperClean Inc.	30	Milano
A4	Elektron Srl.	20	Torino
A5	Acme Ltd.	30	Bologna

INQUINANTE

CodI	NomeI	Stato	PesoSpec	Città
I1	Anidride solforosa	G	12	Torino
I2	Diossina	G	17	Milano
I3	Ammoniaca	L	17	Venezia
I4	Biossido di azoto	S	14	Torino
I5	Mercurio	S	12	Milano
I6	Acido cloridrico	G	19	Torino

PRODOTTO

CodP	NomeP	Città
P1	Detersivo	Milano
P2	Vernice	Venezia
P3	Batteria	Bologna
P4	Detergente	Bologna
P5	Collante	Torino
P6	Nastro magnetico	Firenze
P7	Trasformatore	Torino

SCARICO

CodA	CodI	CodP	Qtà
A1	I1	P1	200
A1	I1	P4	700
A2	I3	P1	400
A2	I3	P2	200
A2	I3	P3	200
A2	I3	P4	500
A2	I3	P5	600
A2	I3	P6	400
A2	I3	P7	800
A2	I5	P2	100
A3	I3	P1	200
A3	I4	P2	500
A4	I6	P3	300
A4	I6	P7	300
A5	I2	P2	200
A5	I2	P4	100
A5	I5	P5	500
A5	I5	P7	100
A5	I6	P2	200
A5	I1	P4	100
A5	I3	P4	200
A5	I4	P4	800
A5	I5	P4	400
A5	I6	P4	500

Esercizi

- Visualizzare il nome e il peso specifico degli inquinanti prodotti nella città di Torino
- Visualizzare tutte le informazioni relative agli inquinanti prodotti nella città di Torino
- Visualizzare tutte le città in cui viene prodotto un inquinante
- Visualizzare il nome delle aziende con sede a Milano e grado di pericolosità superiore o pari a 20
- Visualizzare le coppie di aziende ed inquinanti tali che entrambi si trovino nella medesima città
- Visualizzare il nome degli inquinanti scaricati durante la lavorazione del prodotto con codice P3

Esercizi

- Visualizzare le coppie di nomi di città tali che un azienda che ha sede nella prima città scarichi nella seconda un inquinante relativo a un qualsiasi prodotto
- Visualizzare le coppie di aziende tali che abbiano sede nella stessa città

```
SELECT A.NomeA, B.NomeA,
 FROM AZIENDA AS A, AZIENDA AS B
WHERE A.Città = B.Città
  AND A.CodA < B.CodA;
```

- Visualizzare i nomi delle aziende che producono inquinanti a Milano, a Venezia, o in entrambe le città

Esercizi

- Visualizzare i nomi delle aziende che scaricano inquinanti a Milano
  - Ripetere l'esercizio usando select annidate
- Visualizzare i nomi delle aziende che non scaricano mercurio

```
SELECT NomeA FROM AZIENDA
WHERE CodA NOT IN
    (SELECT CodA FROM SCARICO, INQUINANTE
     WHERE SCARICO.CodI = INQUINANTE.CodI
      AND NomeI = "Mercurio");
```


L'operazione INSERT

- Inserisce un insieme di tuple in una relazione
- Le tuple da inserire possono essere date esplicitamente:

```
INSERT INTO Conto_corrente
<4, "Conti", "V. Piave, 9", 0>,
<5, "Russo", "V. Zara, 19", 100.000>
```

- Oppure le tuple da inserire possono essere il risultato di una interrogazione:

```
INSERT INTO Conto_corrente
SELECT *
FROM Altri_conti
WHERE Saldo > 150.000
```


L'operazione DELETE

- Cancella da una relazione le tuple corrispondenti ad una data condizione
- Esempio:

```
DELETE FROM Movimento
WHERE Causale = "S"
```

L'operazione UPDATE

- Modifica tutte le tuple che soddisfano il predicato dato nel modo indicato
- Ad es. per incrementare del 1 per mille il saldo dei conti correnti con saldo > 100 milioni:

```
UPDATE Conto_corrente
SET Saldo = Saldo*1.001
WHERE Saldo > 100.000.000
```

- Se si vuole aggiornare solo una tupla alla volta, occorre che il predicato sia espresso sugli attributi che compongono una chiave, e che sia soddisfatto per un'unica combinazione di valori:

```
UPDATE Conto_corrente
SET Saldo = Saldo*1.07
WHERE Num_CC = 123
```


Le operazioni per definire lo schema

- Finora abbiamo supposto che le relazioni Conto_corrente e Movimento fossero già presenti nel DB. Vediamo i comandi SQL per crearle

```
CREATE TABLE Conto_corrente
(Num_CC: integer,
 Nome: char(20),
 Indirizzo: char(20),
 Saldo: integer)
CREATE TABLE Movimento
(Num_CC: integer,
 Data_mov: date,
 Num_mov: integer,
 Importo: integer,
 Causale: char(1))
```


Progettazione di un DB

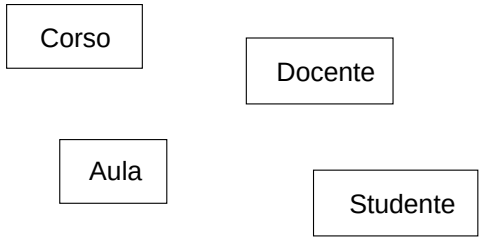
- Progettare un DB vuol dire essenzialmente definirne lo schema
  - capire quali tabelle servono
  - definire lo schema di ciascuna tabella
  - creare le chiavi
- La progettazione delle query e delle modalità di interazione fa invece parte della progettazione delle applicazioni

Procedura di progettazione di un DB

- Innanzitutto si cerca di produrre un *modello concettuale* dei dati da gestire
  - Il modello è concettuale in quanto prescinde dal modello logico dei dati adottato dal particolare DBMS scelto
- Un formalismo molto usato per la descrizione concettuale dei dati è costituito dai diagrammi Entity/Relationship (Chen)
  - *Entità:* ciò che è di interesse per il sistema
  - *Relazioni:* legami di diversa natura tra entità
  - *Attributi:* caratteristiche (proprietà) delle entità e delle relazioni

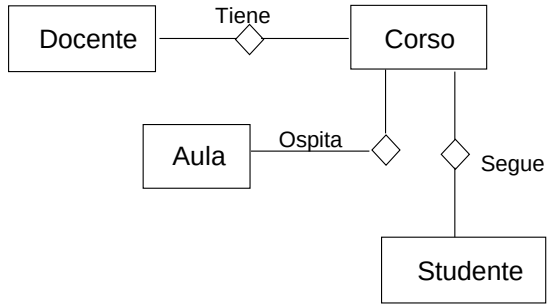
Diagrammi E/R: esempio

- Un sistema per la gestione di corsi di aggiornamento professionale
- Entità:



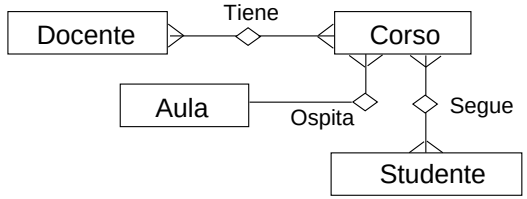
Diagrammi E/R: esempio

- Relazioni:



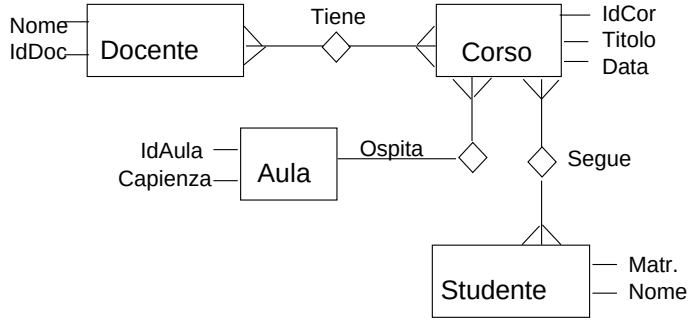
Diagrammi E/R: esempio

- Molteplicità delle relazioni: indicata graficamente dal "ventaglio" ad una estremità della relazione; ad es:
- Aula--<-Corso, indica che una entità Aula può ospitare più Corsi ma un Corso può essere ospitato in una sola aula (relazione uno-molti)
- Corso->---<-Studente indica che una entità Studente può seguire più Corsi, e viceversa un Corso può essere seguito da più studenti (relazione molti-molti)



Diagrammi E/R: esempio

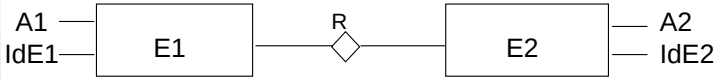
- Attributi:



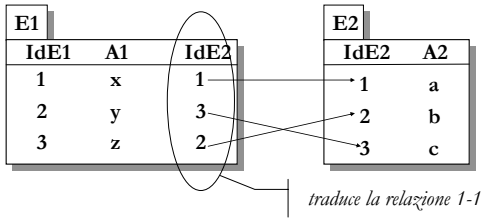
Progettazione di un DB a partire dai diagrammi E/R

- Un diagramma E/R può essere "trasformato" in uno schema logico di DB con qualunque data model
- Tuttavia, i diagrammi E/R sono particolarmente adatti al progetto di DB relazionali, perché la trasformazione di un diagramma E/R in uno schema relazionale è molto semplice:
  - un'entità diventa una relazione (tabella)
  - un attributo di un'entità diventa un attributo di una relazione
  - le relazioni tra entità possono diventare riferimenti diretti da una tupla di una relazione a tupla/e di un'altra oppure possono dar luogo ad una relazione aggiuntiva, a seconda dei casi

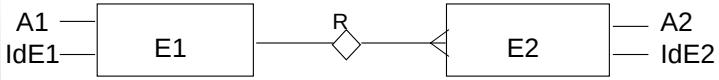
Traduzione di relazioni 1-1



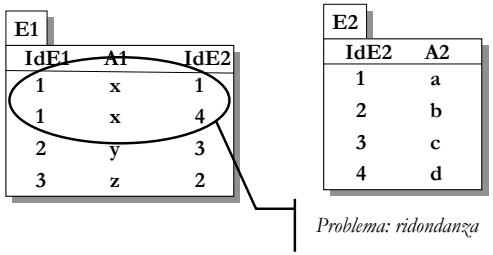
- E1(IdE1: integer, A1: char(1), IdE2: integer)
- E2(IdE2: integer, A2: char(1))



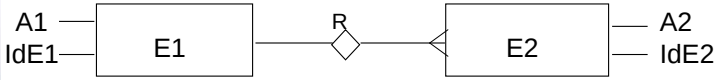
Traduzione di relazioni 1-n: Errore



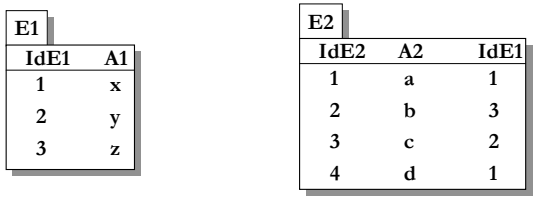
- E1(IdE1: integer, A1: char(1), IdE2: integer)
- E2(IdE2: integer, A2: char(1))



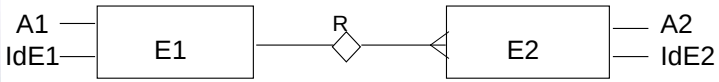
Traduzione di relazioni 1-n: Ok



- Si possono evitare le ridondanze rappresentando la relazione inversa



Traduzione di relazioni 1-n: Alternativa



- Il problema si risolve una volta per tutte introducendo una tabella apposta per la relazione
  - R(IdE1: integer, IdE2: integer)

E1

IdE1	A1
1	x
2	y
3	z

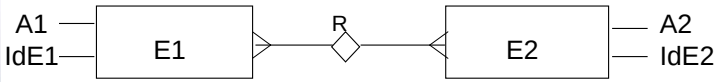
E2

IdE2	A2
1	a
2	b
3	c
4	d

R

IdE1	IdE2
1	1
1	4
2	3
3	2

Traduzione di relazioni n-n



- Anche in questo caso si introduce una tabella apposta per la relazione
  - R(IdE1: integer, IdE2: integer)

E1

IdE1	A1
1	x
2	y
3	z

E2

IdE2	A2
1	a
2	b
3	c
4	d

R

IdE1	IdE2
1	1
1	4
2	3
2	4
3	2